

**21st International Conference in Central Europe
on
Computer Graphics, Visualization and Computer Vision**

in co-operation with

EUROGRAPHICS Association

WSCG 2013

Full Papers Proceedings

Edited by

Manuel M.Oliveira, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil
Vaclav Skala, University of West Bohemia, Czech Republic

**21st International Conference in Central Europe
on
Computer Graphics, Visualization and Computer Vision**

in co-operation with

EUROGRAPHICS Association

WSCG 2013

Full Papers Proceedings

Edited by

Manuel M.Oliveira, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil
Vaclav Skala, University of West Bohemia, Czech Republic

Vaclav Skala – Union Agency

WSCG 2013 – Full Papers Proceedings

Editor: Vaclav Skala
c/o University of West Bohemia, Univerzitni 8
CZ 306 14 Plzen
Czech Republic
skala@kiv.zcu.cz <http://www.VaclavSkala.eu>

Managing Editor: Vaclav Skala

Published and printed by:
Vaclav Skala – Union Agency
Na Mazinách 9
CZ 322 00 Plzen
Czech Republic <http://www.UnionAgency.eu>

Hardcopy: *ISBN 978-80-86943-74-9*

WSCG 2013

International Program Committee

Benes, Bedrich (United States)	Pan, Rongjiang (China)
Benger, Werner (United States)	Paquette, Eric (Canada)
Bengtsson, Ewert (Sweden)	Patow, Gustavo (Spain)
Bilbao, Javier,J. (Spain)	Pedrini, Helio (Brazil)
Biri, Venceslas (France)	Platis, Nikos (Greece)
Bittner, Jiri (Czech Republic)	Reshetov, Alexander (United States)
Buehler, Katja (Austria)	Richardson, John (United States)
Coquillart, Sabine (France)	Rojas-Sola, Jose Ignacio (Spain)
Daniel, Marc (France)	Santos, Luis Paulo (Portugal)
de Geus, Klaus (Brazil)	Savchenko, Vladimir (Japan)
de Oliveira Neto, Manuel Menezes (Brazil)	Skala, Vaclav (Czech Republic)
Debelov, Victor (Russia)	Slavik, Pavel (Czech Republic)
Feito, Francisco (Spain)	Sochor, Jiri (Czech Republic)
Ferguson, Stuart (United Kingdom)	Sourin, Alexei (Singapore)
Gain, James (South Africa)	Sousa, A.Augusto (Portugal)
Gudukbay, Ugur (Turkey)	Sramek, Milos (Austria)
Guthe, Michael (Germany)	Stroud, Ian (Switzerland)
Herout, Adam (Czech Republic)	Szecs, Laszlo (Hungary)
Choi, Sunghee (Korea)	Teschner, Matthias (Germany)
Chover, Miguel (Spain)	Theussl, Thomas (Saudi Arabia)
Chrysanthou, Yiorgos (Cyprus)	Tokuta, Alade (United States)
Juan, M.-Carmen (Spain)	Vitulano, Domenico (Italy)
Kim, HyungSeok (Korea)	Wu, Shin-Ting (Brazil)
Klosowski, James (United States)	Wuensche, Burkhard,C. (New Zealand)
Max, Nelson (United States)	Wuethrich, Charles (Germany)
Molla, Ramon (Spain)	Zara, Jiri (Czech Republic)
Muller, Heinrich (Germany)	Zemcik, Pavel (Czech Republic)
Murtagh, Fionn (United Kingdom)	Zitova, Barbara (Czech Republic)

WSCG 2013

Board of Reviewers

Agathos, Alexander	Fuenfzig, Christoph	Kurt, Murat
Assarsson, Ulf	Gain, James	Kyratzi, Sofia
Ayala, Dolors	Galo, Mauricio	Larboulette, Caroline
Backfrieder, Werner	Gobron, Stephane	Lee, Jong Kwan Jake
Barbosa, Joao	Grau, Sergi	Liu, Damon Shing-Min
Barthe, Loic	Gudukbay, Ugur	Lopes, Adriano
Battiato, Sebastiano	Guthe, Michael	Loscos, Celine
Benes, Bedrich	Hansford, Dianne	Lutteroth, Christof
Benger, Werner	Haro, Antonio	Maciel, Anderson
Bilbao, Javier,J.	Hasler, Nils	Mandl, Thomas
Biri, Venceslas	Hast, Anders	Manzke, Michael
Birra, Fernando	Hernandez, Benjamin	Marras, Stefano
Bittner, Jiri	Hernandez, Ruben Jesus Garcia	Masia, Belen
Bosch, Carles	Herout, Adam	Masood, Syed Zain
Bourdin, Jean-Jacques	Herrera, Tomas Lay	Max, Nelson
Brun, Anders	Hicks, Yulia	Melendez, Francho
Bruni, Vittoria	Hildenbrand, Dietmar	Meng, Weiliang
Buehler, Katja	Hinkenjann, Andre	Mestre, Daniel,R.
Bulo, Samuel Rota	Chaine, Raphaelle	Methodiev, Nikolay Methodiev
Cakmak, Hueseyin	Choi, Sunghye	Meyer, Alexandre
Camahort, Emilio	Chover, Miguel	Molina Masso, Jose Pascual
Casciola, Giulio	Chrysanthou, Yiorgos	Molla, Ramon
Cline, David	Chuang, Yung-Yu	Montrucchio, Bartolomeo
Coquillart, Sabine	Iglesias, Jose,A.	Morigi, Serena
Cosker, Darren	Ihrke, Ivo	Muller, Heinrich
Daniel, Marc	Iwasaki, Kei	Munoz, Adolfo
Daniels, Karen	Jato, Oliver	Murtagh, Fionn
de Geus, Klaus	Jeschke, Stefan	Okabe, Makoto
de Oliveira Neto, Manuel	Jones, Mark	Oyarzun, Cristina Laura
Menezes	Juan, M.-Carmen	Pan, Rongjiang
Debelov, Victor	Kämpe, Viktor	Papaioannou, Georgios
Drechsler, Klaus	Kanai, Takashi	Paquette, Eric
Durikovic, Roman	Kellomaki, Timo	Pasko, Galina
Eisemann, Martin	Kim, H.	Patane, Giuseppe
Erbacher, Robert	Klosowski, James	Patow, Gustavo
Feito, Francisco	Kolcun, Alexej	Pedrini, Helio
Ferguson, Stuart	Krivanek, Jaroslav	Pereira, Joao Madeiras
Fernandes, Antonio	Kurillo, Gregorij	Peters, Jorg

Pina, Jose Luis
Platis, Nikos
Post, Frits,H.
Puig, Anna
Rafferty, Karen
Renaud, Christophe
Reshetouski, Ilya
Reshetov, Alexander
Ribardiere, Mickael
Ribeiro, Roberto
Richardson, John
Rojas-Sola, Jose Ignacio
Rokita, Przemyslaw
Rudomin, Isaac
Sacco, Marco
Salvetti, Ovidio
Sanna, Andrea
Santos, Luis Paulo
Sapidis, Nickolas,S.
Savchenko, Vladimir
Seipel, Stefan
Sellent, Anita

Shesh, Amit
Sik-Lanyi, Cecilia
Sintorn, Erik
Skala, Vaclav
Slavik, Pavel
Sochor, Jiri
Sourin, Alexei
Sousa, A.Augusto
Sramek, Milos
Stroud, Ian
Subsol, Gerard
Sundstedt, Veronica
Szecsi, Laszlo
Teschner, Matthias
Theussl, Thomas
Tian, Feng
Tokuta, Alade
Torrens, Francisco
Trapp, Matthias
Tytkowski, Krzysztof
Umlauf, Georg
Vasa, Libor

Vergeest, Joris
Vitulano, Domenico
Vosinakis, Spyros
Walczak, Krzysztof
WAN, Liang
Wu, Shin-Ting
Wuenske, Burkhard,C.
Wuethrich, Charles
Xin, Shi-Qing
Xu, Dongrong
Yoshizawa, Shin
Yue, Yonghao
Zalik, Borut
Zara, Jiri
Zemcik, Pavel
Zhang, Xinyu
Zhao, Qiang
Zheng, Youyi
Zitova, Barbara
Zwettler, Gerald

WSCG 2013

Full Papers Proceedings

Contents

	Page
Steiger,M., Lücke-Tieke,H., May,T., Kuijper,A., Kohlhammer,J.: Using Layout Stitching to create Deterministic Local Graph Layouts	1
Costa,V., Pereira,M.J., Jorge,A.J.: Fast Compression of Meshes for GPU Ray-Tracing	10
Smith,A.H., Lee,J.K., Hu,H., Mandell,E.S.: Hough Transform-based Technique for Automated Carbon Nanocone Segmentation	19
Phan,A.-C., Raffin,R., Daniel,M.: Joining Meshes with a Local Approximation and a Wavelet Transform	29
Nguyen,H.M., Wuensche,B.C., Delmas,P., Lutteroth,C., Mark,W., Zhang,E.: High-Definition Texture Reconstruction for 3D Image-based Modelling	39
Pribyl,J., Zemcik,P., Burian,T., Kudlac,B.: A Motion-aware Data Transfers Scheduling for Distributed Virtual Walkthrough Applications	49
Plumed,R., Company,P., Varley,P.A.C.: Metrics of human perception of vanishing points in perspective sketches	59
Grau,S., Puig,A., Escalera,S., Salamo,M., Amoros,O.: Efficient complementary viewpoint selection in volume rendering	69
Manya,N., Bengler,W., Ritter,M., Ayyala,I.A., Justic,D., Wang,L.: Cartographic Rendering of Elevation Surfaces using Procedural Contour Lines	79
Seylan,N., Ergun,S., Öztürk,Ö.: BRDF Reconstruction Using Compressive Sensing	88
Alcon,J., Travieso,D., Larboulette,C.: Influence of Dynamic Wrinkles on the Perceived Realism of Real-Time Character Animation	95
Flechon,E., Zara,F., Damiand,G., Jaillet,F.: A generic topological framework for physical simulation	104
Jahrman,K., Wimmer,M.: Interactive Grass Rendering Using Real-Time Tessellation	114
Scheiblauer,C., Wimmer,M.: Analysis of Interactive Editing Operations for Out-of-Core Point-Cloud Hierarchies	123
Sousa,D.J., Cardoso,M.A., Bisch,P.M., Lopes,F.J.P., Travençolo,B.A.N.: A Segmentation Method for Nuclei Identification from Sagittal Images of Drosophila melanogaster Embryos	133
Doyle,R.B., Semwal,S.K.: Computational Celtic Canvas for Zoomorphs and Knotworks	143
Bogoni,T.N., Pinho,M.S.: Haptic Technique for Simulating Multiple Density Materials and Material Removal	151
Saupin,G., Roussel,O., Le Garrec,J.: Robust and Scalable Navmesh Generation	161
Schick,E., Herbort,S., Wöhler,C.: Single view single light multispectral object segmentation	171
Yamamoto,T., Okabe,M., Onai,R.: Semi-Automatic Synthesis of Videos of Performers Appearing to Play User-Specified Music	179

Krolla,B., Bleser,G., Cui,Y., Stricker,D.: Representing Feature Location Uncertainties in Spherical Images	187
Akinci,G., Akinci,A., Oswald,E., Teschner,M.: Adaptive Surface Reconstruction for SPH using 3-Level Uniform Grids	195
Amann,J., Weber,B., Wuethrich,C.A.: Using Image Quality Assessment to Test Rendering Algorithms	205
Moustakas,K.: Handling haptics as a stand-alone medium	215
Bertini,F., Morigi,S.: Deformation by Discrete Elastica	223
Selver,M.A., Ozdemir,M., Selvi,E.: Interactive Radial Volume Histogram Stacks for Visualization of Kidneys from CT and MRI	233
Walek,P., Jan,J., Ourednicek,P., Skotakova,J., Jira,I.: Methodology for Estimation of Tissue Noise Power Spectra in Iteratively Reconstructed MDCT Data	243

Using Layout Stitching to create deterministic Local Graph Layouts

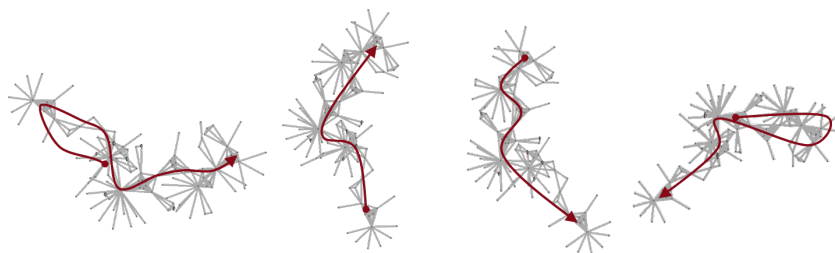
Martin Steiger
Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt
Germany
martin.steiger@
igd.fraunhofer.de

Hendrik
Lücke-Tieke
Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt
Germany
hatieke@
igd.fraunhofer.de

Thorsten May
Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt
Germany
thorsten.may@
igd.fraunhofer.de

Arjan Kuijper
Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt
Germany
arjan.kuijper@
igd.fraunhofer.de

Jörn
Kohlhammer
Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt
Germany
joern.kohlhammer@
igd.fraunhofer.de



ABSTRACT

Dynamic graph layouts are often used to position nodes in local views of large graphs. These layouts can be optimized to minimize changes when navigating to other parts of the graph. Dynamic graph layout techniques do not, however, guarantee that a local layout is recognizable when the user visits the same area twice. In this paper we present a method to create stable and deterministic layouts of dynamic views of large graphs. It is based on a well-known panorama-stitching algorithm from the image processing domain. Given a set of overlapping photographs it creates a larger panorama that combines the original images. In analogy to that our algorithm stitches pre-computed layouts of subgraphs to form a larger, single layout. This deterministic approach makes structures and node locations persistent which creates identical visual representations of the graph. This enables the user to recognize previously encountered parts and to decide whether a certain part of a dataset has already been explored before or not.

Keywords

dynamic graph, explorative analysis, mental map, graph layout stitching

1 INTRODUCTION

Showing the structure emerging from the network connections is one goal of graph visualization. However, human's visual intelligence can be used only if adequate data displays are provided. Using node-link diagrams is a popular visualization technique that works particularly well for small to medium-sized graphs. The goal of our technique is to ease the exploration of local

structures of large static graphs based on a node-link diagram.

A typical user task is the exploration of the neighborhood around a focal area of the graph. We consider such an exploration as success if the user is able to mentally chart the visible parts of the graph. Thus, it increases the area the user is familiar with. To be precise, "familiarity" reflects two abilities to us: Firstly, the user is able to recall a visible area upon revisiting. Secondly, the user is able to mentally extend this area beyond the visible part enabling the user to plan and predict navigation. Revisitation has been identified by Lee et al. [LPP⁺06] as one of the tasks to be supported by graph visualization.

The tasks we support with our technique are characterized by the following two assumptions: The first one is

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

that the information on the local level is more important than the global structure of the graph. The second assumption is that movement along the edges is required to gather required information. An example for such tasks are investigations in citation networks or social networks. We derive two conflicting requirements from

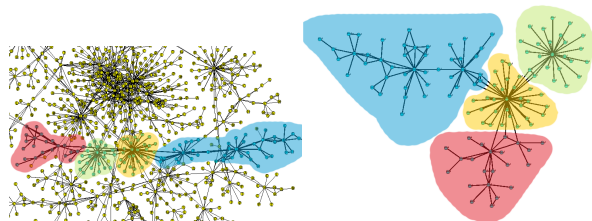


Figure 1: Identical network parts have been highlighted in a global layout (left) and a local, independent layout (right). The former display makes the impression that the clusters were linearly connected. In particular, it seems as if the shortest path from red to blue led through green and yellow. However, the local view reveals that this is not the case.

these goals and task characteristics. In the following, we will describe these requirements, show why they are conflicting and propose a solution to resolve this conflict as the core contribution of this paper:

The first requirement is to show only the part of the graph the user is interested in and to adapt the layout to this visible subgraph. To give an example, Figure 1 shows a global layout with five clusters in a linear arrangement. Instead, a local layout of the clusters exposes the actual topology of these clusters. This requirement is dealt with in a number of existing *dynamic layout* approaches. Dynamic layout covers techniques for the selection of interesting or important areas of the graph, for the definition of incremental layouts and the smooth transition between consecutive layouts of these visible subgraphs during exploration. We will call these visible subsets of the graph *frames* from now on. The rationale for smooth transition is to minimize the users' effort to keep track of the evolving layout. However, this applies to consecutive frames only. Revisiting a known area of the graph after extensive exploration usually results in very different layouts.

This fact leads to our second requirement. Whenever a user revisits an area of the graph for a second time the difference between the two layouts should be as small as possible. We state that this requirement applies regardless of the length or direction of the user's exploration path between any two visits of the same area. For static graphs we argue that the visible layout is determined by the currently visible subgraph only. A simple solution exists if only the second requirement were to be considered: Compute a *static* layout of the complete graph and toggle the visibility of nodes and edges as needed. However, a static layout conflicts with the first

requirement, because it naturally does not adapt to local features.

Our solution is a resolution of this conflict fulfilling both requirements. We propose a dynamic layout that adapts to the currently visible subgraph, but which is independent of the exploration path. The main challenge is to keep the layouts "stable" during exploration. We use a two-level-layout strategy to solve this problem. The first-level layout is done before interactive exploration: We compute a set of overlapping subgraphs which covers the entire graph. The overlapping cover guarantees that most of the nodes will appear in at least two subgraphs. For every subgraph a layout is computed independently to produce the first-level layout. These layouts serve as "building-blocks" for the second level layout that is used during exploration.

The challenge of the second-level layout is to combine these layouts depending on a given visible area of the graph. The node coordinates from different subgraph layouts need to be merged in a deterministic fashion. To achieve this goal we choose a technique that originally comes from the field of image processing: *Panorama stitching* is used to merge a set of overlapping photographs into a single, seamless image. We transfer this technique to graphs. Depending only on the currently visible frame, individual subgraph layouts are selected, weighted and merged to produce the final, visible layout.

To the best of our knowledge, these conflicting requirements have not been solved with a single technique before. Our contribution is a technical solution serving as a proof-of-concept which fulfills both requirements. It extends the notion of layout stability from "frame-to-frame-coherence" to "frame consistency". This means that the layout of any subgraph looks the same or at least similar for every visit. As a concept, it makes use of existing approaches to create subgraphs and their layouts, but it is not bound to specific approaches. In fact, we believe that this concept offers a design space, which is worth to be explored further in future.

The rest of the paper is organized as follows: Section 2 discusses related work in the area, before the concept of our method is described in Section 3. In Section 4 we present test results for artificial and real datasets before we conclude with discussion and future work in Section 5.

2 RELATED WORK

In this section we first describe fundamental work on the preservation of the mental map for the navigation in network visualization, especially with respect to design considerations and criteria. After that, we present techniques which have been developed to tackle this problem by improving layouts and/or interaction.

The preservation of the mental map of graph visualization has become an important goal ever since its introduction by Eades et al. [ELMS91]. In the literature, dynamic layout techniques have been proposed to solve two different problems: The first problem is the layout computation of dynamic graphs, which has been formalized by North [Nor96]. The second problem is the layout of a dynamic *view* of a graph, which has been described by Huang et al. [HEW98]. A dynamic view is basically a visible subgraph, which can be “moved” interactively for browsing. This problem has been applied to static graphs, for example, by Huang et al. [HEL05] and van Ham and Perer [vHP09]. According to North layout stability is achieved by minimizing layout changes between consecutive frames. Interestingly, the two problems are similar from the perspective of this criterion alone. In fact, many existing techniques could be used to solve both problems. Our definition for layout stability, however, does not apply to consecutive frames alone. In addition we require that the layout of any given subgraph will be the same upon revisitation. Hence we can only claim to solve the second problem here; the dynamic view of a static graph.

We consider the layout stability as a means to augment recall on recently visited regions of the graph. The results of Marriott et al. [MPWG12] suggest that layout features have different cognitive impact, e.g. favoring symmetry or orthogonality. In an earlier study, Purchase and Samra [SP08] note that minimal node movement may not be the most relevant criterion for mental map preservation. Archambault et al. [APP11] compare animation approaches to small-multiple approaches, but their effect on mental map preservation are inconclusive. We have to note that especially recall experiments are naturally limited to small graphs - and schemes to transfer results to real world graphs have yet to be devised.

Many dynamic layout techniques are modifications of static layout techniques which impose specific constraints or quality objectives on the transition between two consecutive layouts. Brandes and Mader [BM12] compare different measures, especially with respect to the trade-off between individual layout quality and stability between frames. They note that even slightly lowered requirements in quality often offer a significant boost in stability. Virtually all elements of a graph visualization have been covered by previous approaches to stabilize the mental model upon dynamic changes. For example, Frishman and Tal [FT08] and Erten et al. [EHK⁺04] propose approaches where quality objectives apply to the node movement. Frishman and Tals approach fine-tunes the inertia of nodes between consecutive frames. Erten et al. propose a natural extension to force-feedback techniques by using a (2+1)-dimension layout using virtual edges connecting different time-frames. Other approaches, like that of Dwyer

et al. [DMS⁺08] and Frishman and Tal’s [FT04] focus on the preservation of node clusters in the dynamic layout. Additionally, Dwyer’s approach optimized the arrangement of polyline-edges. Aside from spring-embedding layouts, dynamic layout methods have also been used in conjunction with other techniques. For example, Görg et al. [GBPD05] use Sugiyama-style layout techniques.

Among these approaches, our technique relates most to cluster-preserving dynamic layouts. However, the “clusters” in our approach are subgraphs, which are laid-out independently in a preprocessing step and merged together depending on the current area of interest of the user. Archambault et al. [AMA07] present a similar strategy with the static multi-level technique *Topolayout*. Topolayout creates hierarchical partition layouts with the most suited technique and merges them to minimize edge lengths and crossings. In contrast, our technique creates overlapping subgraphs which are dynamically merged along the overlapping nodes.

Aside from techniques which aim to preserve the mental map on a purely structural level, the role of interaction and navigation cues must be considered as well. In fact, Marriott et al. note in their study [MPWG12] that node labels are more powerful cues for mental mapping. However, we think that layout stability supports the effective use of local navigation cues like labels, because they need to be located in the view to be useful. Moscovich et al. [MCH⁺09], van Ham and Perer [vHP12] and May et al. [MSDK12] propose techniques to ease navigation across larger distances. Their common idea is to provide visual cues pointing to otherwise invisible nodes or regions of the graph.

3 CONCEPT

In this section we will describe how we derive a deterministic global layout from a set of local layouts. We therefore transfer the panorama stitching algorithm to the graph layout domain. Before we can perform our layout stitching algorithm, a set of subgraphs with overlapping node sets needs to be created. A local layout is then computed for every subgraph - independent from the rest of the graph. We refer to these subgraph layouts as *patches* from now on. We then align these patches to match the positions of all nodes that exist in more than one subgraph as good as possible. The position of nodes that exist in multiple patches are then merged and a unified layout is created. The basic idea is illustrated in Figure 2. In the final step, we will explain how to create a layout that consists of more than just two patches.

3.1 Definitions

We define a graph $G(V, E)$ comprising a set of vertices V together with a set of edges E . Our method works

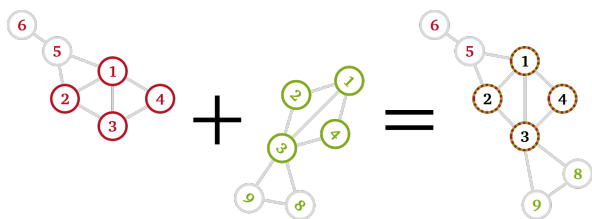


Figure 2: The green layout patch is aligned to the red patch using the four shared nodes. Nodes that exist in both layouts are merged, creating a unified layout of both patches.

with both directed and undirected edges without limitations. However, we will assume for the sake of simplicity that the graph is connected, i.e. a path exists between every pair of nodes in G . We also define a clustering $C(V)$ as a mapping of V to a set of classes, so that every vertex in G is linked to one or more classes.

3.2 Pre-processing

The series of visible frames is defined by the user who is browsing this graph on a local level. We do not define the means of interaction here, but we assume that the set of visible nodes can be derived from the user's interaction. Our concept defines a local layout for this subgraph. Given a set of visible nodes, the set of layout patches that need to be merged can be derived. The frame which was displayed during the last timestep does not influence the layout of the current frame. This ensures that the same picture is created – independent from the exploration path.

If no set of patches for a graph is provided, we compute a cover of overlapping subgraphs from a topology-based clustering, so that every vertex of the graph is contained in at least one patch. Our approach works with basically any clustering algorithm. However, we point out that the cluster size and content have an influence on their layout which in turn influences the cluster shape that is used for the stitching.

The resulting clusters cannot be used directly as patches, because the clustering typically creates a partition of the graph, i.e. every node is contained in exactly one single cluster only. A straight-forward approach to make them overlap is to include neighbors of the first degree. In other words: nodes from other patches that are directly connected are added to the node set of the patch. Larger sets are created by adding neighbors of neighbors and so on.

We consider two clusters to be connected if they share at least three nodes. This is the minimum number of points that is required for the layout patch alignment computation.

As soon as the subgraphs are created, a layout is computed for each of them. This can be performed completely independently which allows for using

different layout algorithms. Moreover, the computation can be done in a pre-processing step, but also deferred until the layout is actually needed which avoids unnecessary computational overhead. Force-directed algorithms such as that of Fruchterman and Reingold appear to be a sound choice as they reveal local structure and are flexible to integrate user-specified requirements[Kob12].

3.3 Shape Matching

The sub-layouts are computed independently, therefore the position of nodes is given in a local coordinate system. Nodes that exist in more than one layout generally have different positions in each of them. We will now describe how two patches with overlapping node sets can be aligned so that the distance in between is minimized. Individual node positions do not fit perfectly, but this will be fixed in a later step.

The idea of stitching shapes is based on the work of Brown and Lowe[BL07] who describe an approach for automatic panorama stitching. The authors compute a matching transformation for images based on distinct, but overlapping point clouds. This process is far less complex for graph layouts as no image post-processing such as brightness compensation is required. Most importantly, the point correspondences in the two point sets is known in our setting which simplifies the algorithm.

The second, important contribution comes from the the shape-matching algorithm of Müller et al. which works with identical point clouds but in a very different context[MHTG05]. The authors present a method that allows for elastic deformation of three-dimensional objects. With the help of shape matching, the points of the deformed object can be gradually transformed back to their original position. For that, the two geometric point sets are compared and a transformation that reduces the pair-wise distance between all points to a minimum is deduced. Apart from translation and scaling, their transformation scheme offers refinements such as twisting and compression which are not present in the work of Brown and Lowe.

We trivially acquire a set of vertices that exist in two given layouts by computing the intersection of the two sets.

As long as the set contains at least three vertices, we can use the standard least-squares fitting method [AHB87] to compute a deterministic matching transformation. For the sake of simplicity, we will restrict this computation to rigid transformation, i.e. rotation and translation, but general affine transformations are feasible as well. For every point p in sublayout A we specify its counterpart p' in sublayout B as

$$p'_i = Rp_i + t + \varepsilon_i$$



Figure 3: Rotating one of the two point clouds (green) reduces the average distance between pairs.

Here, R is a rotation matrix, t a translation vector and ε the measure of error. After solving for ε and accumulating the error over all points, we get

$$\varepsilon^2 = \sum_{i=1}^n \|\varepsilon_i\|^2 = \sum_{i=1}^n \|p'_i - (Rp_i + t)\|^2$$

The error becomes minimal if both point clouds have the same centroid [AHB87]. This can be achieved by subtracting the centroid of their respective sets (denoted as c_p and $c_{p'}$) from the point locations. The task is now to find an optimal rotation matrix where the pairwise distance is minimal for all points. This matrix can be deduced from a 2×2 cost matrix H that measures the distance between two point clouds. We subtract the centroids from both datasets to bring them to the origin and define this matrix H as

$$H = \sum_i^n (p_i - c_p)(p'_i - c_{p'})^T$$

Using singular value decomposition (SVD), the matrix H can be factorized into two rotations U and V and a diagonal scaling matrix S .

$$[U, S, V] = SVD(H)$$

See, for example, the introduction by Wall et al. [WRR03] for details on the mathematical background of the singular value decomposition. The final desired rotation matrix can be computed as:

$$R = VU^T$$

The final result is a transformed point \hat{p}_i that represents the point p_i of sublayout A in the coordinate system of sublayout B .

$$\hat{p}_i = R(p_i - c_p) + c_{p'}$$

The accumulated difference between \hat{p}_i and the original point p'_i relates to the previously computed error ε and can be used as a quality measure for this transformation process.

3.4 Combining multiple shape matching transformations

The approach we presented so far works well for combining two patches. However, in general, a frame consists of more than that. We therefore describe how to

stitch multiple patches in one frame and how we create a smooth transition between two consecutive frames. The problem we solve here is to find a deterministic order in which the patches are stitched. Therefore, we create a meta-graph of the patches. Two patches are connected, if two patches share common nodes (see Figure 5 left and center). Thus, they can be stitched together. For the remaining part of the paper, this graph is referred to as *patch graph*.

If the patches that should be merged are connected directly, only a deterministic order of stitching operations needs to be defined. Otherwise, also a connecting series of patch stitchings must be generated to ensure that also distant patches can be combined.

This series of stitchings of overlapping patches can be seen as a path in the patch graph. Also, on this level of abstraction, the interactive exploration can be seen as a user-driven traversal of this patch graph.

Starting with a single patch, the user continues exploring, eventually reaching a part of the graph, that can not be visualized without including additional patches in the visible subgraph. Adjacent patches are then added until the requested graph region can be visualized. This graph traversal must be stateless and therefore independent of previously visible patches. If this was not the case, different exploration paths would have different stitching orders thus result in different global layouts. Three possible setups are depicted in Figure 4.

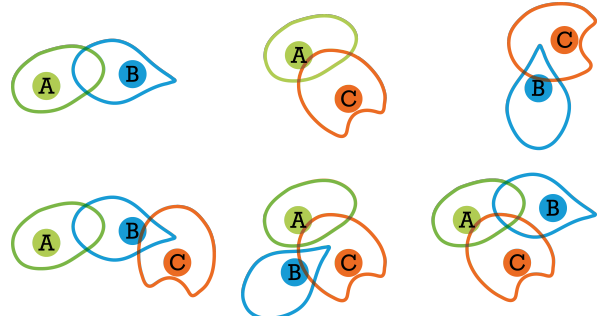


Figure 4: Three patches (A, B, C) with corresponding 1-to-1 stitchings (top row). If the patches would be stitched in the order they become visible, different stitched layouts would result. In this configuration three different global layouts could be produced (bottom row).

The reason for this is that the patch graph contains multiple paths that connect the visible patches. Reducing the number of edges naturally leads to a reduction of the number of paths. To enforce a stable matching order, we remove all edges from the patch graph that are not strictly necessary to keep the graph connected (see the illustrations in Figure 5). What is left is a spanning tree of the graph and can be computed by Kruskal's algorithm.

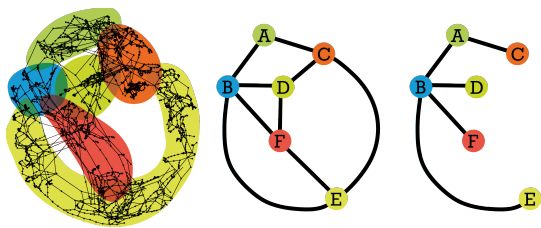


Figure 5: The original graph is reduced to a graph of patches which is then reduced to a spanning tree. This tree is used to define unique paths between any two nodes.

It is also able to incorporate edge weights, thus computing the spanning tree with lowest total weight – the minimum spanning tree (MST). Starting with a graph that contains all nodes but no edges, edges with the lowest weight are continuously added as long as they don't lead to cycles in the graph.

Although the error value of each matching seems like a natural choice to maximize the quality of the whole layout, several drawbacks lead us to the decision against using it. First and foremost, using the matching error as edge weight is possible only if the matching error was known for all pairs of connected patches. Computing the optimal affine transformation of all possible combinations of layout patches is rather time-consuming. Furthermore, interactive manipulation of a single patch layout would result in changing weights for its incident edges, which in turn could cause changes in the spanning tree of the patch graph.

Instead, we define a similarity-based weight function so that edges between pairs of patches with large overlap ratios have lower edges weights. They are then most likely to be stitched first. The Jaccard similarity coefficient is a measure that indicates how similar two sets are and is defined as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

This measure does not require the computation of the matching error between all edges in the patch graph. Similar to the distance function that is often used in graph clustering, we can use this (or another) similarity measure and assign this value to the edges of the patch graph.

For every pair of layout patches in this spanning tree, only one single path exists. These paths in the tree are no longer the shortest in general when compared to the original patch graph, but this reduction in freedom results in consistent patch stitching chains. This also ensures independence of previous frames, as the stitching order is fixed. We use the root of the spanning tree as end point for all paths. Thus, every visible patch is stitched to its parent patch until the root node is encountered. This is a critical aspect as it ensures that patches

are always matched to the same neighbor patches. The local position of a node is thus transformed by the series of affine transformations of the patches along the path to the root patch.

Some nodes belong to multiple patches and would, without additional correction, have multiple positions on the drawing canvas. We therefore derive from all these positions a commonly shared, unique position. In such cases, we use a linear combination of the nodes' weight factor to place the nodes depending on time and the user focus. This ensures a smooth transition from one layout frame to another.

4 PRELIMINARY TESTS

In this section we will present some test results of both artificial and real datasets. First we demonstrate the concept in detail using a basic test graph. Second, we use a real dataset to demonstrate that different exploration paths result in congruent layouts.

4.1 Concept verification

The first test run is based on a graph of the form of a Venn diagram for three sets (see Figure 6). It contains three node rings that overlap at the center. This graph is small yet complex enough to test the correctness of our approach. Its structure allows, on the one hand, the extraction of three overlapping patches – the rings – and ensures, on the other hand that their layouts overlap only very little while having excellent matching error scores.

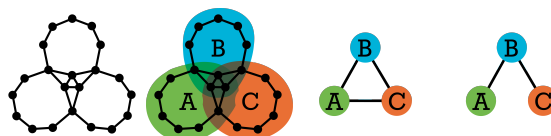


Figure 6: From left to right: The Venn diagram (1) is split into three overlapping subgraphs (2). These form a patch graph (3) with 3 patches and 3 edges which is reduced to a tree (4) to enable a stable interactive exploration.

We create the patch graph and compute a spanning tree. Using the tree, we then merge one patch after another in coordination with the interactive exploration component. The green patch is shown first and thus forms the root patch for rendering and remains as it is (Figure 7 left). The blue patch is flipped, rotated and translated to the bottom of the green, minimizing the matching error between the green and the blue patch. The common nodes are then merged, creating the layout in Figure 8 center). In the next step, the orange patch is aligned with the blue, already aligned patch. This results in a total transformation (Figure 8 right) of about 180° for the orange patch. Lastly, the node positions are unified where necessary and merged for the final, visible graph.



Figure 7: Individual layouts of the three patches of the Venn diagram graph.

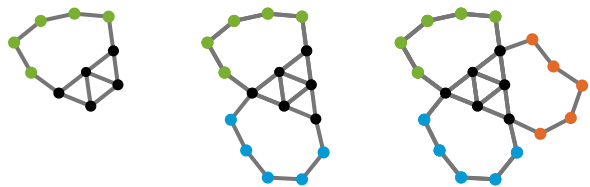


Figure 8: Initially, only the green layout is visible (left). In the next step, the blue patch is matched against the green patch (center). Finally, the orange patch is matched to the blue patch (right).

The second test we performed was with a pair of star-shaped subgraphs which has been extracted from a real dataset. The layout of both subgraphs is strongly affected by the high degree of the central nodes. The intersection of the node sets contains only four elements, but both star nodes are included. This leads to a significant overlap of the patches, but the star patterns are still visible (Figure 9).

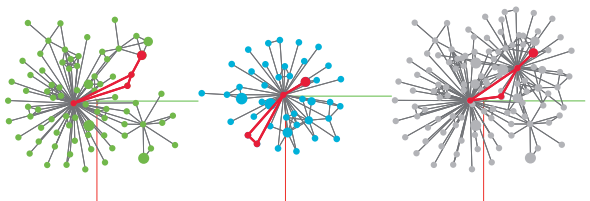


Figure 9: Two star-shaped patches (green, blue) are stitched together forming a single graph layout (gray). The common nodes (red) that form the base for the matching are the only nodes that are distorted. Although the central nodes are in both sets, both patches are recognizable in the stitched layout.

In a similar dataset, two star-shaped graphs have been extracted again, but this time, they intersect only at their boundaries.

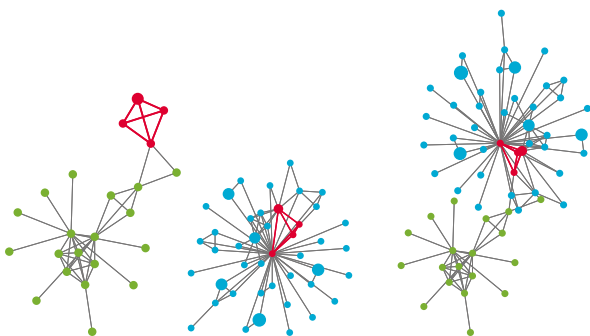


Figure 10: Two star-shaped patches (green, blue). A clear outlier region in the green patch leads to a clear separation in the stitched layout (right).

As can be seen in Figure 10, the patterns are stitched with only very little deformation of the original shapes. More importantly, the nodes that form the connection between the two clusters are clearly distinguishable in the stitched layout.

4.2 Exploration independence

The claim of this paper is to create a deterministic layout which is independent of the exploration path. We test this hypothesis by navigating through several clusters of a larger graph in different order and compare the generated layouts.

The dataset for this test is a network graph from the medical domain [GCV⁺07] with roughly 1.5k nodes, 5.5k edges. Our clustering algorithm created 77 layout patches. The size of the graph features a fair amount of complexity while still being visually comprehensible when viewed as a whole (Figure 11). We used a force-directed layout of the whole dataset to display the exploration path as ground truth and compare the results. With only one single parameter – the number of iterations – the *chinese whispers* algorithm [Bie06] appeared to be a good choice for the clustering of this graph.

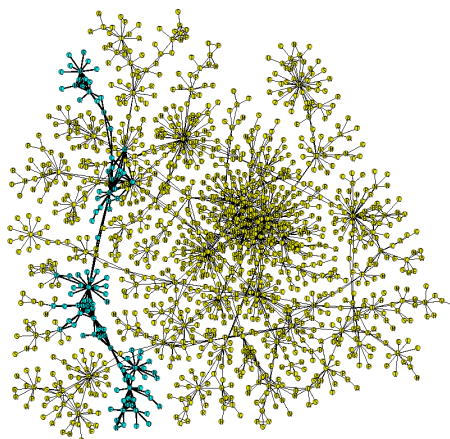


Figure 11: The explored graph part in the global layout

Several different explorations have been performed to verify the validity of our approach. They all started at different points exploring the same clusters, but in different orders. As can be seen in the teaser figure on the first page, the resulting stitched layouts are congruent. The construction of two exemplary stitched layouts is depicted in Figure 12 and Figure 13, respectively.

5 CONCLUSION & OUTLOOK

In this paper we presented a new approach that aims to create dynamic graph layouts which are independent of the exploration path. It works independent of specific layout algorithms and thus also works for highly dynamic force-directed layout algorithms. When the user explores large graphs with dynamic views, new nodes

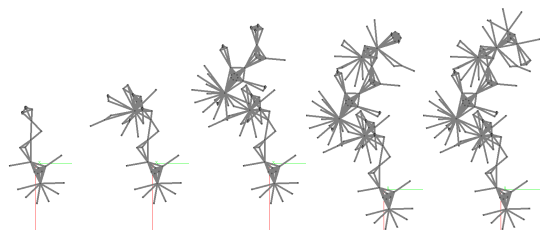


Figure 12: The first exploration through five clusters in the order 1, 2, 3, 4, 5.

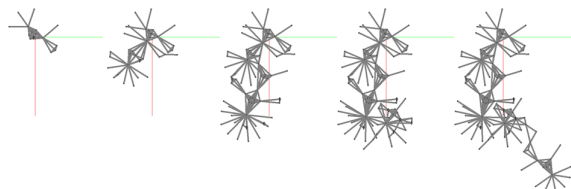


Figure 13: In the second run, the clusters were explored in the inverse order resulting in a congruent layout.

are typically added in the proximity of existing, linked nodes.

This approach is thus highly dependent on the exploration path – the layout can look very different even for very similar explorations. Our method overcomes this limitation with techniques from the computer vision domain where image stitching is used to merge multiple photographs with overlapping areas into a larger image. In analogy to that, our method uses pre-computed layout patches that are sewn together in deterministic order. Consequently, the resulting layout is stable, independent of the user’s exploration path and will, thus, always look the same. In contrast to many other dynamic graph layout algorithms, a fair amount of computational effort can be pre-computed which increases the interactivity and reduces the workload at runtime. Being able to work with different layout algorithms for different patches makes it also very versatile.

Compared to conventional layout methods, the additional computational effort is also rather small. The cost of layout computation is increased by the factor of nodes that exist in multiple matches. Runtime costs are limited to the creation a 2×2 cost matrix and its decomposition which has a constant running time [MHTG05].

The layout stitching method we presented sees the subgraph as a disconnected point cloud and merges the patches without respect to the topological structure. Closely related to that, it also ignores the points that are not in the intersection of the two nodes sets. As a result, two layouts could be aligned so that the disjoint parts overlap as well which is undesired.

We assume that more sophisticated approaches for the computation of the patch overlaps could mitigate this problem and improve the stitching quality. This includes the use of the graph topology metrics such as connectivity to find and include the best-fitting nodes. An ideal strategy would include nodes that emphasize

certain visual features of the cluster layout to make the structure memorable. We are convinced that the interpretation of layouts as images features a plethora of concepts and approaches just waiting to be transferred and applied to graph layouts.

6 REFERENCES

- [AHB87] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-Squares Fitting of Two 3D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, sept. 1987.
- [AMA07] D. Archambault, T. Munzner, and D. Auber. TopoLayout: Multilevel Graph Layout by Topological Features. *Visualization and Computer Graphics, IEEE Transactions on*, 13(2):305–317, march-april 2007.
- [APP11] D. Archambault, H. Purchase, and B. Pinard. Animation, Small Multiples, and the Effect of Mental Map Preservation in Dynamic Graphs. *Visualization and Computer Graphics, IEEE Transactions on*, 17(4):539–552, april 2011.
- [Bie06] Chris Biemann. Chinese Whispers: an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, TextGraphs-1, pages 73–80, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.
- [BL07] Matthew Brown and David G. Lowe. Automatic Panoramic Image Stitching using Invariant Features. *International Journal of Computer Vision*, 74:59–73, 2007.
- [BM12] Ulrik Brandes and Martin Mader. A quantitative comparison of stress-minimization approaches for offline dynamic graph drawing. In Marc Kreveld and Bettina Speckmann, editors, *Graph Drawing*, volume 7034 of *Lecture Notes in Computer Science*, pages 99–110. Springer Berlin Heidelberg, 2012.
- [DMS⁺08] T. Dwyer, K. Marriott, F. Schreiber, P. Stuckey, M. Woodward, and M. Wybrow. Exploration of networks using overview+detail with constraint-based cooperative layout. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1293–1300, nov.-dec. 2008.
- [EHK⁺04] Cesim Erten, Philip J. Harding, Stephen G. Kobourov, Kevin Wampler,

- and Gary Yee. GraphAEL: Graph Animations with Evolving Layouts. In Giuseppe Liotta, editor, *Graph Drawing*, volume 2912 of *Lecture Notes in Computer Science*, pages 98–110. Springer Berlin Heidelberg, 2004.
- [ELMS91] Peter Eades, Wei Lai, Kazuo Misue, and Kozo Sugiyama. Preserving the mental map of a diagram. *Proceedings of COM-PUGRAPHERS*, 91(9):24–33, 1991.
- [FT04] Y. Frishman and A. Tal. Dynamic Drawing of Clustered Graphs. In *IEEE Symposium on Information Visualization (InfoVis '04)*, pages 191–198, 2004.
- [FT08] Y. Frishman and A. Tal. Online Dynamic Graph Drawing. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):727–740, july-aug. 2008.
- [GBPD05] Carsten Görg, Peter Birke, Mathias Pohl, and Stephan Diehl. Dynamic Graph Drawing of Sequences of Orthogonal and Hierarchical Graphs. In János Pach, editor, *Graph Drawing*, volume 3383 of *Lecture Notes in Computer Science*, pages 228–238. Springer Berlin Heidelberg, 2005.
- [GCV⁺07] Kwang-II Goh, Michael E. Cusick, David Valle, Barton Childs, Marc Vidal, and Albert-László Barabási. The Human Disease Network. *Proc. of the National Academy of Sciences USA*, 104(21):8685–8690, 2007.
- [HEL05] Xiaodi Huang, Peter Eades, and Wei Lai. A framework of filtering, clustering and dynamic layout graphs for visualization. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science - Volume 38, ACSC '05*, pages 87–96, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.
- [HEW98] Mao Lin Huang, Peter Eades, and Junhu Wang. Online Animated Visualization of Huge Graphs using a Modified Spring Algorithm. *Journal of Visual Languages & Computing*, 9(6):623–645, 1998.
- [Kob12] Stephen G. Kobourov. Spring embedders and force directed graph drawing algorithms. *CoRR*, abs/1201.3011, 2012.
- [LPP⁺06] Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, BELIV '06, pages 1–5, New York, NY, USA, 2006. ACM.
- [MCH⁺09] Tomer Moscovich, Fanny Chevalier, Nathalie Henry, Emmanuel Pietriga, and Jean-Daniel Fekete. Topology-aware navigation in large networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 2319–2328, New York, NY, USA, 2009. ACM.
- [MHTG05] Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Meshless Deformations Based on Shape Matching. *ACM Trans. Graph.*, 24(3):471–478, July 2005.
- [MPWG12] K. Marriott, H.C. Purchase, M. Wybrow, and C. Goncu. Memorability of Visual Features in Network Diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2477–2485, dec. 2012.
- [MSDK12] T. May, M. Steiger, J. Davey, and J. Kohlhammer. Using Signposts for Navigation in Large Graphs. *Computer Graphics Forum*, 31(3 pt. 2):985–994, 2012.
- [Nor96] Stephen C. North. Incremental layout in dynadag. In *Proceedings of the Symposium on Graph Drawing*, GD '95, pages 409–418, London, UK, UK, 1996. Springer-Verlag.
- [SP08] Peter Saffrey and Helen Purchase. The "mental map" versus "static aesthetic" compromise in dynamic graphs: a user study. In *Proceedings of the ninth conference on Australasian user interface - Volume 76, AUIC '08*, pages 85–93, Darlinghurst, Australia, Australia, 2008. Australian Computer Society, Inc.
- [vHP09] Frank van Ham and Adam Perer. Search, Show Context, Expand on Demand: Supporting Large Graph Exploration with Degree-of-Interest. *IEEE Transactions on Visualization and Computer Graphics*, 15:953–960, 2009.
- [vHP12] Frank van Ham and Adam Perer. Integrating Querying and Browsing in Partial Graph Visualizations. *IBM Technical Report 12-01*, 2012.
- [WRR03] Michael Wall, Andreas Rechtsteiner, and Luis Rocha. Singular Value Decomposition and Principal Component Analysis. *A practical approach to microarray data analysis*, pages 91–109, 2003.

Fast Compression of Meshes for GPU Ray-Tracing

Vasco Costa
INESC-ID/IST
Rua Alves Redol, 9
1000-029 Lisboa,
Portugal
vasco.costa@ist.utl.pt

João M. Pereira
INESC-ID/IST
Rua Alves Redol, 9
1000-029 Lisboa,
Portugal
jap@inesc-id.pt

Joaquim A. Jorge
INESC-ID/IST
Rua Alves Redol, 9
1000-029 Lisboa,
Portugal
jaj@inesc-id.pt

ABSTRACT

We present a novel and expedite way to compress triangles meshes, fans and strips for ray-tracing on a GPU. Our approach improves on the state of the art by allowing the lossless compression of all connectivity information without changing the mesh configuration, while using linear time and space with the number of primitives. Furthermore, the algorithm can be run on a stream processor and any compressed primitive can be indexed in constant time, thus allowing fast random-access to geometry data to support ray-tracing on a GPU. Furthermore, both triangle and quad meshes compress particularly well, as do many type-specialized mesh structures where all primitives have an equal number of vertexes. Our results show that the compression algorithm allows storing and ray-tracing meshes with tens of millions of triangles on commodity GPUs with only 1GB of memory.

Keywords

Ray-tracing, gpu, mesh, compression.

1 INTRODUCTION

Polygon meshes are the most common representation for scene geometry. Triangles are the most common primitive although quad meshes are also popular in some applications being particularly suitable for architectural scenes where large and flat surfaces are the most preminent features in a scene.

Triangle meshes may also be represented with triangle fans or triangle strips. These allow additional space savings by taking advantage of the fact that there will often be common edges in a triangular mesh. The common edges in fans and strips also mean the computational costs required to compute visibility will be reduced by virtue of symmetry in the adjacent triangles.

However, meshes are a very inefficient and storage consuming way of representing complex scenes, which makes it difficult to fit even moderately complex scenes inside graphical cards with limited memory, in more complex scenes it may be required to use different kinds of primitive types to enable higher face data compression as can be observed in Figure 1.

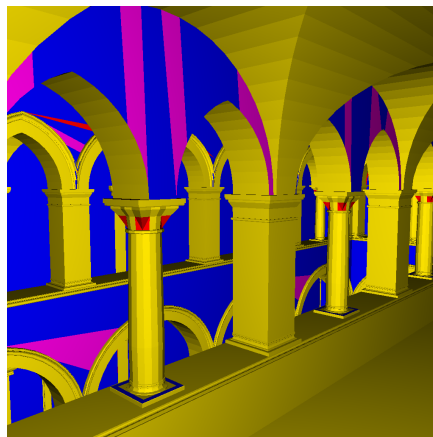


Figure 1: Sponza scene. *Triangles* are shown in red, *quads* are yellow, *triangle fans* are blue, *triangle strips* are violet. A triangle mesh representation would use 1.62 MB of space compared to the representation in the figure which only requires 1.17 MB. Thus we achieve a 72% compression ratio just by employing these more complex primitives.

A common way used to store such meshes with different primitive types is displayed in a simplified form in Figure 2. For example in PBRT [PH10] a scene is stored in a list of primitives where each primitive is subclassed from a main object class. This leads to much waste of memory storing pointers, C++ class data, etc when the scene is a mesh, so PBRT supports type specialized triangles meshes as well for improved performance in such cases as can be seen in Figure 4. Similar special-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

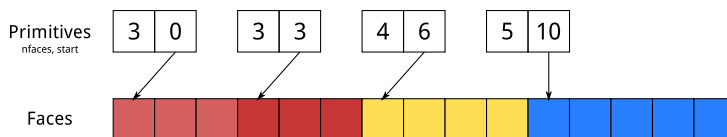


Figure 2: Regular data structure to store an n-gon mesh.

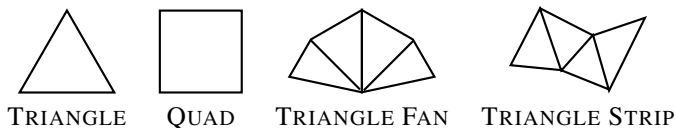


Figure 3: Primitive types.

ized quad meshes could have been implemented just as well as can be observed in Figure 5.

In this work we shall present an algorithm for mesh storage. This algorithm shall enable storing triangle meshes and quad meshes with low storage requirements, like the type specialized versions, thanks to an innovative way of storing and compressing the primitive array data using arithmetic encoding. In addition the algorithm can also store and compress n-gon meshes with triangles, quads, triangle fans, and triangle strips. Face data is stored in a compressed array with the leading zeros trimmed out.

Our algorithm thus employs non-lossy primitive compression (arithmetic encoding) and face compression (discard leading zeros). It can also quantize 32-bit vertex coordinate data down to 16-bits in a lossy fashion. In practice the lossy compression scheme seems to have little impact on final output quality for the tested scenes, as can be seen in Table 4, and enhances compression further. In order to minimize the loss of precision all coordinates are converted from world to scene coordinates prior to the quantization step.

Our algorithm *does not require expensive preprocessing*, e.g. the construction of temporary data structures for doing adjacency queries on the mesh, so it runs in $O(n)$ linear time. It produces similar compression results to other more complex hybrid geometry and acceleration structure compression schemes.

Finally we will do a performance comparison of our achieved compression ratios versus the industry standard GZIP [Deu96] compression tool which uses a Lempel-Ziv [ZL77] compression scheme. GZIP is inherently serial since it is required to read previous values to determine the next consecutive value in the stream.

We note that our algorithm features constant time $O(1)$ random access to any primitive, while GZIP works only

on streaming data, thus GZIP requires $O(n)$ time in the worst case to access any random primitive. Hybrid schemes often store the scene in a tree structure in which accesses take $O(\log n)$ time to complete.

2 RELATED WORK

We are going to limit ourselves to mentioning other algorithms which work purely on scene compression first. Our algorithm is intended for generic use, specifically for meshes, and is agnostic to the kind of ray tracing acceleration scheme being used.

For those rendering algorithms which operate on large blocks of data say on a page level basis, such as out-of-core algorithms, they may still find Lempel-Ziv or other similar general purpose compression schemes worthwhile. While these algorithms are inherently serial multiple blocks can be worked in parallel with a minor compression ratio penalty. This is the approach followed by the LZSS streaming compression algorithms [OSC12]. Also essential is work on processing variable length data on streaming architectures [Bal10] in an efficient fashion with a parsimonious use of atomic operations.

Given that we are using a ray-tracer primitive intersection tests for triangles [MT97], quads [LD05], fans [GA05], and triangle meshes in general [AC97] demand being mentioned.

In the realm of geometric compression and mesh optimization several works stand out:

- Isenburg [ILS05a, ILS05b] uses arithmetic encoding to compress vertex coordinates while taking advantage of parallelogram predictors in triangular meshes by virtue of having knowledge beforehand of the mesh topology leading to improvements in the encoding predictor function.
- Yoon [YLP05, YL06] reorders the geometry in order to increase memory coherency during the rendering pass thus improving rendering performance.

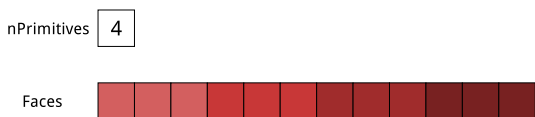


Figure 4: Specialized triangle mesh.

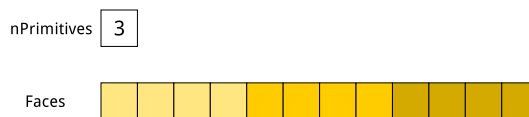


Figure 5: Specialized quad mesh.

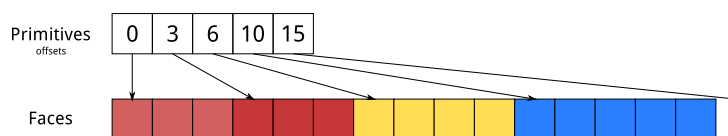


Figure 6: Compact data structure to store an n-gon mesh.

There are other interesting works on compressing ray tracing acceleration structures together with the geometry. These are not acceleration scheme agnostic and take advantage of local knowledge coming from the cells. For example, take a partition cell's bounding box, and improve compression of both the partitioning structure and the geometry contained therein since you know the vertices range of values is constrained to the box.

Most of the interest in this sort of scheme lies with bounding volume hierarchies (BVHs) since in that scenario you do not need to store the same faces twice in different leaves of the tree.

BVHs do not have duplicated primitive instances in different cells. Yoon [YL07, KMKY10] has done a lot of work in this area of hybrid BVH and mesh storage schemes. More recently Garanzha [GBG11] has worked on a more simplified hybrid BVH and mesh storage scheme implemented for use on a streaming architecture. Another approach is that taken by Lauterbach [LYM07] where a hybrid Kd-tree and triangle strip scheme is used to represent mesh data. The main issue with all of these particular hybrid algorithms is their complexity, requiring expensive preprocessing e.g. the construction of temporary data structures for doing adjacency queries on the mesh, and their difficulty of implementation. Preprocessing takes a long time so these methods are not useful for dynamic scenes with destructible geometry. The encoding methods of Garanzha are much more amenable for GPU implementation than Yoon's more elaborate work and the algorithm provides good rendering performance due to the use of a surface area heuristic (SAH) BVH, good data locality, and aligned memory loads.

Our work is intended to be *acceleration structure agnostic* so we do not rely on any of these schemes. The algorithm we devised is also amenable to implementation on a streaming architecture and can be computed in $O(n)$ linear time. Our algorithm may be used on any object/space subdivision structure: BVHs, KD-trees, Grids. It is also applicable for other applications which do not require the use of an acceleration structure and just require random access to the mesh geometry.

3 ALGORITHM

Our algorithm compresses an n-gon scene. As an example the scene can be described in the .OBJ file format.

3.1 Scene Loading

The scene loader processes the scene data and generates a regular data structure such as that seen in Figure 2 as output.

Since .OBJ file format n-gons are not necessarily planar this means we cannot use explicit ray-polygon intersection routines safely.

So we load the scene n-gons in the following fashion:

- if the polygon has *three* faces, the output is a *triangle* which is stored in the primitive and face arrays.
- if the polygon has *four* faces, the output is a *triangle fan* which is stored in the primitive and face arrays.
- if the polygon has *five or more* faces, the output is passed through the GLUTESSELATOR which splits the n-gon into *triangles*, *triangle fans*, and *triangle strips* that are in turn stored in the primitive and face arrays.

In the next step we process an array such as the one in Figure 2 into a compact array like the one which can be seen in Figure 6. This is done with a SCAN operation:

```
def scan(uint *prims, uint nprims) {
  for (uint i=1; i<=nprims; ++i) {
    prims[i+1] += prims[i];
  }
}
```

Listing 1: SCAN.

The complexity of a SCAN also known as PREFIX-SUM operation is $O(n)$ but in a parallel processor it can be computed even faster. With such a pass we reduce the amount of memory required to store the primitive array by roughly a half.

3.2 Geometry Compression

Next to the primitive array compaction we proceeded to its compression.

We employ arithmetic encoding to compress the primitive array using the PACKPRIMITIVES function. The predictor function we are employing assumes all the primitives in the scene have the same number of faces. So for triangle meshes and quad meshes the primitive array is shrunk to nothing and the array degenerates to those seen in Figures 4 and 5 respectively. This is the optimum outcome.

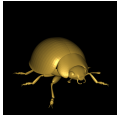

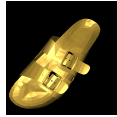
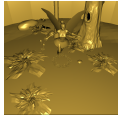
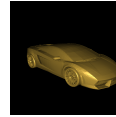
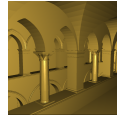
						
	LADY BIRD	BULLDOZER	SANDAL	FAIRY FOREST	LAMBORGHINI	SPONZA
num vertices	23903	105507	2636	97124	575416	39742
original						
num faces	93984	436587	11676	365949	3017847	149926
num triangles	0	145529	1197	17715	1005949	1170
num quads	23496	0	1970	78201	0	34819
num fans	0	0	29	0	0	649
num strips	0	0	0	0	0	248
triangulated						
num faces	140976	436587	15852	522351	3017847	228462
num triangles	46992	145529	5284	174117	1005949	76154

Table 1: Scene geometry data.

For more complex scenes with dissimilar primitives the delta between the predicted and actual value is stored packed tightly as a small integer of range 2^{pMSB} in a bit array.

```
def packPrimitives(uint *prims, uint nprims) {
    float avg;
    avg = float (prims [ nprims + 1 ])/ nprims ;

    int Min = 0;
    int Max = 0;

    min = prims [ nprims + 1 ];
    for (uint i=0; i<=nprims; i++) {
        uint predict = avg*i;
        uint actual = prims [ i ];
        int diff = long (actual)-predict;

        Min = min (diff, Min);
        Max = max (diff, Max);
    }

    // arithmetic encode
    pMSB = log2 (Max-Min);
    hprims = callocBits (nprims+1, pMSB);

    for (uint i=0; i<=nprims; i++) {
        uint predict = avg*i;
        uint actual = prims [ i ];

        uint diff = actual-predict-Min;
        pack (hprims, i, diff);
    }

    uint2 params;
    params . x = as_int (avg);
    params . y = Min;
    return params, hprims;
}
```

Listing 2: PACKPRIMITIVES.

Compression of face data proceeds in a similar fashion to the small integer packing method mentioned before. We compress away the leading zeros in the face data using the PACKFACES function.

The small integers will have a range of 2^{fMSB} . The faces of primitives other than triangles are shifted to the

left one bit to accommodate a flag stating if they are a triangle fan (0) or triangle strip (1) respectively.

```
def packFaces(uint *faces, uint nfaces, uint nvertices) {
    uint fMSB = log2 (nvertices * 2);
    hfaces = callocBits (nfaces, fMSB);

    // compress leading zeros
    for (uint i=0; i<nfaces; ++i) {
        pack (hfaces, i, faces [ i ]);
    }
    return hfaces;
}
```

Listing 3: PACKFACES.

Finally we quantize the vertexes from 32-bits per x, y, z component to 16-bits per component. First we take care to ensure we do this while operating in scene bounding box space in order to reduce the range of data we will compress. Then the quantization to 16-bits per component is done. This is our only lossy compression step. This PACKVERTICES function compresses vertexes by 50%.

```
def packVertices(Axisbox bounds, uint *vertices, uint nvertices) {
    hvertices = new ushort3 [ nvertices ];

    // transform to scene bounding box coordinates and quantize to 16-bits
    const float3 scale = 65535.0f / (bounds . Max - bounds . Min);

    for (uint i=0; i<nvertices; ++i) {
        hvertices [ i ] = vertices [ i ] - bounds . Min * scale;
    }
    return hvertices;
}
```

Listing 4: PACKVERTICES.

The error produced by the quantization step is minimal in the scenes we tested as can be seen on the rightmost column in Table 4.

In a typical scene composed of manifolds there are more polygons than vertexes so face compression is very important contrary to what one might otherwise assume. This can easily be established empirically by looking at the scenes in Table 1.

3.3 Intersection Testing

During ray tracing scene traversal it will be required to test if a given primitive is intersected by a ray. All primitive intersection tests are done using the Möller-Trumbore [MT97] ray-triangle intersection algorithm.

This can be accomplished with the TEXTPRIMITIVE pseudo-code.

```
bool testPrimitive(Axisbox bounds, uint id, Intersection *isect) {
    const uint i = prims[id];
    const uchar nverts = prims[id+1] - i;

    switch (nverts) {
    case 3:
    {
        const uint3 idx = vload3(0, faces+i);
        return testTriangle(bounds, idx, vertices, ray, isect);
    }
    break;
    case 4:
    {
        const uint4 idx = vload4(0, faces+i);
        return testQuad(bounds, idx, vertices, ray, isect);
    }
    break;
    default:
    {
        if ((faces[i] & 1)) {
            return testStrip(bounds, nverts, vertices, faces+i, ray, isect);
        } else {
            return testFan(bounds, nverts, vertices, faces+i, ray, isect);
        }
    }
    break;
    }
    return false;
}
```

Listing 5: TESTPRIMITIVE.

Unfortunately, like we mentioned before, it cannot be trusted that the quads in an .OBJ file will be planar as is indeed the case in the FAIRY FOREST, SANDAL, and other test scenes. It is debatable if we should just constrain the primitives to e.g. triangles and fans in order to support n-gons since it would greatly simplify the control code and save 1 bit per face.

4 TEST METHOD

The algorithm was implemented in C++ and OpenCL on Linux. The test platform is an AMD FX 8350 8-core CPU @ 4.0 GHz powered machine with 8 GB of RAM. The graphics card includes a NVIDIA GeForce GTX 660 Ti GPU with 2 GB of RAM. All ray tracing rendering is offloaded to the GPU.

The ray-tracing engine supports hashed grids [LD08] as a spatial subdivision acceleration structure. All the compute intensive parts of the grid construction algorithm are also run on the GPU. Due to limitations of space we do not describe this engine in detail here.

All scenes were rendered at 1024×1024 resolution with one sample per pixel using Phong shading.

We selected several test scenes not just for having large amounts of geometry but for the richness of their polygon soup so to speak. In order of presentation the scenes are:

LADY BIRD	Quad mesh of an organic creature.
BULLDOZER	Triangle mesh of a construction vehicle.
SANDAL	More complex primitives. e.g. triangle fans in the sole of the shoe.
FAIRY FOREST	Standard benchmark scene which features both triangles and quads.
LAMBORGHINI	Large triangle mesh of a car with over 1M triangles.
SPONZA	Complex scene in terms of geometric detail due to the arches but also features several flat surfaces such as walls.

For more information on the scene geometry data please consult Table 1.

The following tests were considered to be interesting:

- To test our proposed compression methods vs other schemes, namely [GBG11], specific for geometry compression for GPU ray tracing purposes.

- To determine the performance of the primitive array predictor function in the arithmetic coding phase we test the misprediction residuals i.e. the delta between the predicted and actual values in the test scenes.

- Compression ratio of our proposed compression methods vs uncompressed data in terms of: space savings and rendering frame rate.

- Compression ratio of our proposed compression methods vs GZIP to determine how good our compression algorithms are at achieving space savings compared to a standard streaming compression algorithm.

5 RESULTS

As can be seen in the left of Table 2 our implementation uses less memory than the non-compressed triangle meshes of [GBG11] because we do not store duplicate triangle vertexes. Our compressed face and vertex scheme achieves similar compression results to their compressed quad mesh for the tested scenes *without* requiring any pre-processing to convert the triangle mesh to a quad mesh as they do. Our algorithm also supports the use of quad meshes but, as can be seen in the right of Table 2, this is not required to achieve good compression ratios due to our face compression algorithm and the storage of unique vertexes only.

One of the big issues with any scheme employing arithmetic coding is getting the right prediction function. In our case, since we want random access in constant time we cannot consult prior values since that would require decoding them beforehand, plus all the previous values to those, to begin with. So our predic-

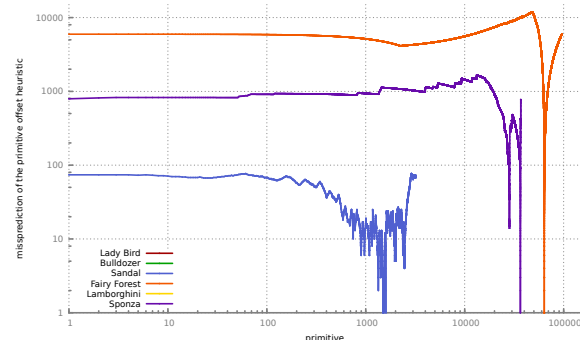


Figure 7: Delta between the predicted and actual values in the test scenes. It displays perfect prediction in the triangle and quad meshes such as Lady Bird, Bulldozer, Lamborghini scenes. For the other scenes the misprediction residuals can be quite large.

	Ours P	[GBG11] NCT	Ours PFV	[GBG11] CQ
DRAGON 7 MTRI	123.92 MB	247.85 MB	80.03 MB	82.62 MB
THAI 10 MTRI	171.66 MB	343.32 MB	114.44 MB	114.44 MB
LUCY 28 MTRI	481.61 MB	963.22 MB	331.11 MB	321.07 MB

Table 2: Comparison of the space required to store a mesh using our algorithm versus Garanzha [GBG11]. First we compare the memory usage of both our specialized triangle meshes. Next we compare the performance of our compressed face and vertex scheme versus their compressed quad mesh.

tion function must rely only on a table of values independent of the compressed array. In our case we use a linear prediction function to approximate the primitive data values. This works well when the scenes all have similar sized n-gons i.e. triangle or quad meshes where the residuals are zero. The mispredictions and residuals increase with the storage of different sized n-gons. This can be observed in Figure 7 where the LADY BIRD, BULLDOZER, LAMBORGHINI primitive arrays get compressed to 0 bits per primitive while on the other scenes namely SANDAL, FAIRY FOREST, and SPONZA this does not happen. This problem could probably be reduced by employing higher order prediction functions such as polynomial functions with more terms than our linear function.

From the extensive test results, which can be observed in Table 3, we managed to confirm several of our hypothesis as matters of fact. The support for triangle and quad mesh scenes, such as LADY BIRD, BULLDOZER, and LAMBORGHINI, is excellent with very good compression capabilities and, in some cases, we even achieve enhanced frame rates over the uncompressed baseline due to improved data locality caused by the compression. This is most obvious when compressing the primitives array and enabling the lossy vertex compression together in the pure triangle and quad meshes i.e. the PV results. This option also enables reasonable compression ratios in the order of $\sim 70\%$.

We can also observe that for the more complex scenes using large n-gons we often get higher frame rates by triangulating the scene beforehand. This is rather evident in the SPONZA and SANDAL scenes in particular. This is due to spatial subdivision. These larger primitives will occupy more cells in the grid and hence there will be more redundant intersection calculations going on. This could perhaps be improved with the use of mailboxing. We did not attempt to use mailboxing in our implementation. Another possible improvement is better specialized ray-triangle fan, and ray-triangle strip intersection routines which reduce the amount of redundant ops such as those mentioned in Section 2. This does not apply to the FAIRY FOREST scene because of its smaller n-gons. That scene consists of only triangles and quads.

We get our worst frame rate results when face compression is enabled. This is due to the loss of use of vectorized memory loads in our implementation and the unaligned memory accesses to access elements after the compression. This can most likely be improved further by creating dedicated vectorized load operations for our bitarray structures.

Invariably the highest compression ratios happen when we enable all of our compression techniques: primitive compression, face compression and vertex compression i.e. the PFV results. In fact with all these techniques enabled we get better compression ratios than GZIP in many scenes such as LADY BIRD, SANDAL, FAIRY FOREST. The only scenes where GZIP manages to win over our compression scheme are those scenes where the vertexes have redundant coordinates or there are redundant vertexes such as the BULLDOZER, the LAMBORGHINI, and SPONZA. This problem with vertex quantization techniques had already been identified by Isenburg et al.

We remind the reader that contrary to GZIP our algorithm allows random access to any primitive in the scene in $O(1)$ constant time which is essential for ray-tracing and other applications. This is particularly relevant for stringy kd-trees with few primitives per leaf. We get compression ratios in the order of $\sim 40 - 50\%$ which is commensurate with streaming lossless compression techniques which are a lot harder to parallelize on a GPU properly since they require sequential access to compress or decompress data.

6 CONCLUSIONS

We have demonstrated an n-gon (triangle, quad, triangle fan, triangle strip) scene compression algorithm which can compress such a scene in linear $O(n)$ time with constant $O(1)$ scene primitive access time. For the special case where the n-gons in the scene are all the same size like triangle or quad meshes the additional space used over a type specialized structure is essentially zero. The algorithm can optionally provide limited compression ratios with improved rendering performance, or much improved compression with worse rendering performance than in the uncompressed case. The compression ratios, in the order of $\sim 40 - 50\%$, are competitive with those achieved using the GZIP tool which, contrary to our algorithm, does not allow constant time random access to the data.

In the future it should be possible to significantly improve the primitive array compression level using non-linear prediction functions. We also believe that either a limit on the size of triangle fans and triangle strips is imposed, in order to improve the performance under spatial subdivision ray tracing, or there needs to be a way to more quickly detect misses for such complex primitives, use mailboxing, or more than one of these

schemes. Otherwise the rendering performance for the complex primitives will be subpar, even if the compressed sizes for scenes using these primitives would be a lot better.

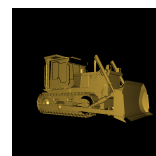
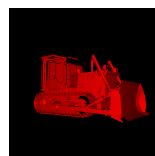
7 ACKNOWLEDGMENTS

This work was supported by national funds through FCT - Fundação para a Ciência e Tecnologia, under project PEst-OE/EEI/LA0021/2013.

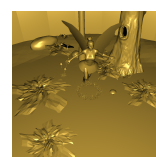
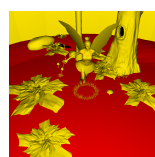
We would like to thank Gilles Tran (Lady Bird), Ralph Garmin (Bulldozer), John Burkardt (Sandal), Ingo Wald (Fairy Forest), Temur Kvitelashvili (Lamborghini), Marko Dabrovic (Sponza) and the Stanford 3D Scanning Repository (Dragon, Thai, Lucy) for the test scenes.

REFERENCES

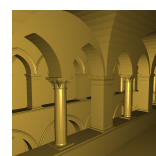
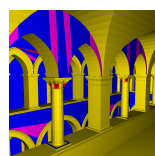
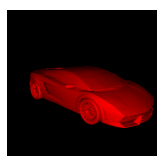
- [AC97] J. Amanatides and K. Choi. Ray Tracing Triangular Meshes. In *Proceedings of the Eighth Western Computer Graphics Symposium*, pages 43–52, 1997.
- [Bal10] A. Balevic. Parallel variable-length encoding on GPGPUs. In *Euro-Par 2009, Parallel Processing Workshops*, pages 26–35. Springer, 2010.
- [Deu96] P. Deutsch. GZIP file format specification version 4.3. RFC 1952, May 1996.
- [GA05] E. Galin and S. Akkouche. Fast Processing of Triangle Meshes using Triangle Fans. In *Shape Modeling and Applications, 2005 International Conference*, pages 326–331. IEEE, 2005.
- [GBG11] K. Garanzha, A. Bely, and V. Galaktionov. Simple Geometry Compression for Ray Tracing on GPU. In *GraphiCon'2011*, 2011.
- [ILS05a] M. Isenburg, P. Lindstrom, and J. Snoeyink. Lossless Compression of Predicted Floating-Point Geometry. *Computer-Aided Design*, 37(8):869–877, 2005.
- [ILS05b] M. Isenburg, P. Lindstrom, and J. Snoeyink. Streaming Compression of Triangle Meshes. In *ACM SIGGRAPH 2005 Sketches*, page 136. ACM, 2005.
- [KMKY10] T. Kim, B. Moon, D. Kim, and S. Yoon. RACBVHs: Random-accessible compressed bounding volume hierarchies. *Visualization and Computer Graphics, IEEE Transactions on*, 16(2):273–286, 2010.
- [LD05] A. Lagae and P. Dutré. An Efficient Ray-Quadrilateral Intersection Test. *Journal of Graphics Tools*, 10(4):23–32, 2005.
- [LD08] A. Lagae and P. Dutré. Compact, Fast and Robust Grids for Ray Tracing. In *Computer Graphics Forum*, pages 1235–1244. Wiley Online Library, 2008.
- [LYM07] C. Lauterbach, S. Yoon, and D. Manocha. Ray-Strips: A Compact Mesh Representation for Interactive Ray Tracing. In *Interactive Ray Tracing, 2007. RT'07. IEEE Symposium on*, pages 19–26. IEEE, 2007.
- [MT97] T. Möller and B. Trumbore. Fast, Minimum Storage Ray-Triangle Intersection. *Journal of Graphics Tools*, 2(1):21–28, 1997.
- [OSC12] A. Ozsoy, M. Swamy, and A. Chauhan. Pipelined Parallel LZSS for Streaming Data Compression on GPGPUs. In *Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on*, pages 37–44. IEEE, 2012.
- [PH10] M. Pharr and G. Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2010.
- [YL06] S. Yoon and P. Lindstrom. Mesh Layouts for Block-Based Caches. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):1213–1220, 2006.
- [YL07] S. Yoon and P. Lindstrom. Random-Accessible Compressed Triangle Meshes. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1536–1543, 2007.
- [YLP05] S. Yoon, P. Lindstrom, V. Pascucci, and D. Manocha. Cache-Oblivious Mesh Layouts. *ACM Transactions on Graphics (TOG)*, 24(3):886–893, 2005.
- [ZL77] J. Ziv and A. Lempel. A Universal Algorithm for Sequential Data Compression. *Information Theory, IEEE Transactions on*, 23(3):337–343, 1977.



LADY BIRD					BULLDOZER				
TECHNIQUES	GZIP	REGULAR	739.03 KB	FRAME RATE	GZIP	REGULAR	3.43 MB	COMP RATIO	FRAME RATE
		TRIANGULATED	1014.37 KB			TRIANGULATED	3.43 MB		
		OURS	COMP RATIO			OURS			
V	393.12 KB	739.03 KB	100 %	102.94 FPS	1.28 MB	3.43 MB	100 %		88.58 FPS
F	...	598.97 KB	81.05 %	100.93 FPS	...	2.82 MB	82.39 %		87.87 FPS
FV	...	555.46 KB	75.16 %	81.44 FPS	...	2.70 MB	78.74 %		69.24 FPS
P	...	415.41 KB	56.21 %	78.06 FPS	...	2.10 MB	61.13 %		68.66 FPS
PV	...	647.24 KB	87.58 %	111.16 FPS	...	2.87 MB	83.81 %		94.51 FPS
PF	...	507.18 KB	68.63 %	104.92 FPS	...	2.27 MB	66.19 %		95.30 FPS
PFV	...	463.68 KB	62.74 %	84.25 FPS	...	2.14 MB	62.55 %		73.26 FPS
		323.62 KB	43.79 %	85.53 FPS		1.54 MB	44.94 %		74.93 FPS
T	465.78 KB	1014.37 KB	100 %	87.16 FPS	1.28 MB	3.43 MB	100 %		87.98 FPS
TV	...	874.31 KB	86.19 %	90.81 FPS	...	2.82 MB	82.39 %		88.40 FPS
TF	...	739.03 KB	72.86 %	65.04 FPS	...	2.70 MB	78.74 %		69.23 FPS
TFV	...	598.97 KB	59.05 %	67.92 FPS	...	2.10 MB	61.13 %		68.52 FPS
TP	...	830.80 KB	81.9 %	97.14 FPS	...	2.87 MB	83.81 %		94.37 FPS
TPV	...	690.74 KB	68.1 %	100.40 FPS	...	2.27 MB	66.19 %		95.62 FPS
TPF	...	555.46 KB	54.76 %	71.23 FPS	...	2.14 MB	62.55 %		73.08 FPS
TPFV	...	415.40 KB	40.95 %	73.98 FPS	...	1.54 MB	44.94 %		74.91 FPS

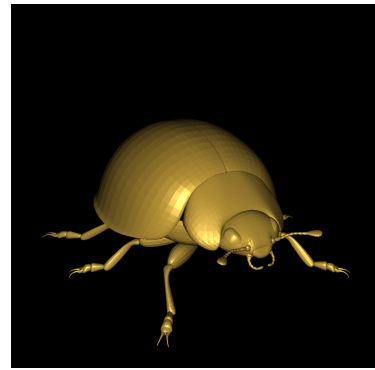
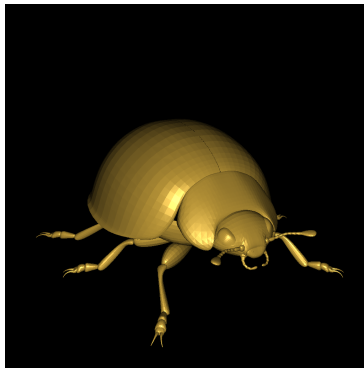
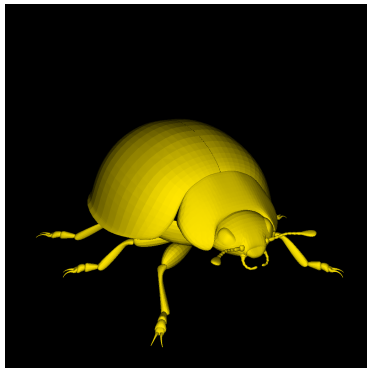


SANDAL					FAIRY FOREST				
TECHNIQUES	GZIP	REGULAR	88.99 KB	FRAME RATE	GZIP	REGULAR	2.87 MB	COMP RATIO	FRAME RATE
		TRIANGULATED	113.46 KB			TRIANGULATED	3.77 MB		
		OURS	COMP RATIO			OURS			
V	54.56 KB	88.99 KB	100 %	195.99 FPS	1.54 MB	2.87 MB	100 %		25.45 FPS
F	...	73.55 KB	82.64 %	186.20 FPS	...	2.32 MB	80.66 %		25.03 FPS
FV	...	61.91 KB	69.57 %	157.39 FPS	...	2.26 MB	78.74 %		19.39 FPS
P	...	46.47 KB	52.21 %	151.69 FPS	...	1.71 MB	59.4 %		18.92 FPS
PV	...	79.23 KB	89.03 %	180.00 FPS	...	2.67 MB	92.84 %		22.52 FPS
PF	...	63.79 KB	71.68 %	168.37 FPS	...	2.11 MB	73.5 %		21.82 FPS
PFV	...	52.15 KB	58.6 %	145.47 FPS	...	2.06 MB	71.58 %		17.18 FPS
		36.71 KB	41.25 %	141.51 FPS		1.50 MB	52.24 %		17.41 FPS
T	61.66 KB	113.46 KB	100 %	193.17 FPS	1.80 MB	3.77 MB	100 %		21.95 FPS
TV	...	98.02 KB	86.39 %	183.96 FPS	...	3.21 MB	85.25 %		21.34 FPS
TF	...	76.69 KB	67.6 %	158.19 FPS	...	2.90 MB	76.87 %		16.90 FPS
TFV	...	61.25 KB	53.98 %	151.65 FPS	...	2.34 MB	62.12 %		16.49 FPS
TP	...	92.81 KB	81.8 %	196.04 FPS	...	3.10 MB	82.37 %		23.72 FPS
TPV	...	77.37 KB	68.19 %	188.65 FPS	...	2.55 MB	67.63 %		23.66 FPS
TPF	...	56.05 KB	49.4 %	158.18 FPS	...	2.23 MB	59.24 %		17.67 FPS
TPFV	...	40.60 KB	35.78 %	154.70 FPS	...	1.68 MB	44.49 %		17.76 FPS

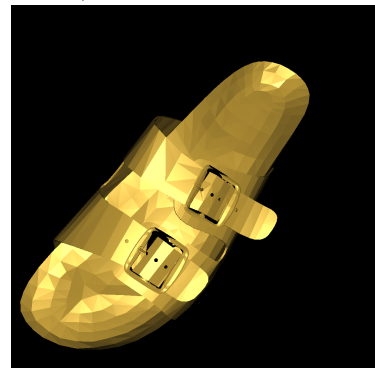
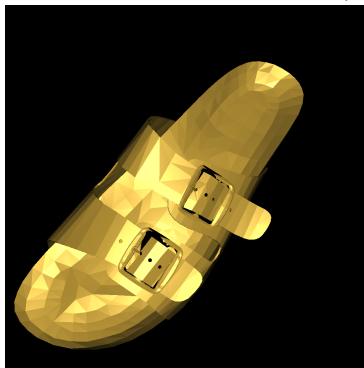
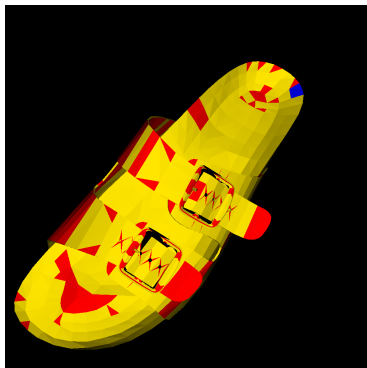


LAMBORGHINI					SPONZA				
TECHNIQUES	GZIP	REGULAR	21.93 MB	FRAME RATE	GZIP	REGULAR	1.17 MB	COMP RATIO	FRAME RATE
		TRIANGULATED	21.93 MB			TRIANGULATED	1.62 MB		
		OURS	COMP RATIO			OURS			
V	8.87 MB	21.93 MB	100 %	53.27 FPS	426.95 KB	1.17 MB	100 %		54.73 FPS
F	...	18.64 MB	84.99 %	54.10 FPS	...	962.61 KB	80.52 %		52.20 FPS
FV	...	17.98 MB	81.96 %	43.74 FPS	...	920.95 KB	77.04 %		43.49 FPS
P	...	14.68 MB	66.95 %	44.19 FPS	...	688.08 KB	57.56 %		41.24 FPS
PV	...	18.10 MB	82.51 %	58.93 FPS	...	1.08 MB	92.09 %		50.47 FPS
PF	...	14.80 MB	67.49 %	61.11 FPS	...	868.04 KB	72.61 %		47.14 FPS
PFV	...	14.14 MB	64.46 %	47.16 FPS	...	826.38 KB	69.13 %		40.10 FPS
		10.85 MB	49.45 %	49.03 FPS		593.52 KB	49.65 %		38.38 FPS
T	8.87 MB	21.93 MB	100 %	53.02 FPS	548.34 KB	1.62 MB	100 %		64.05 FPS
TV	...	18.64 MB	84.99 %	53.77 FPS	...	1.39 MB	85.94 %		59.70 FPS
TF	...	17.98 MB	81.96 %	43.84 FPS	...	1.21 MB	74.73 %		48.51 FPS
TFV	...	14.68 MB	66.95 %	43.93 FPS	...	1004.45 KB	60.67 %		45.88 FPS
TP	...	18.10 MB	82.51 %	58.93 FPS	...	1.33 MB	82.03 %		62.45 FPS
TPV	...	14.80 MB	67.49 %	60.80 FPS	...	1.10 MB	67.97 %		59.45 FPS
TPF	...	14.14 MB	64.46 %	47.14 FPS	...	939.83 KB	56.77 %		47.35 FPS
TPFV	...	10.85 MB	49.45 %	48.96 FPS	...	706.97 KB	42.7 %		45.95 FPS

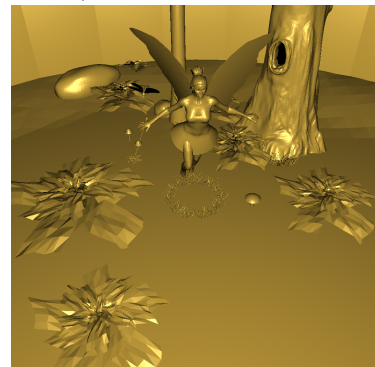
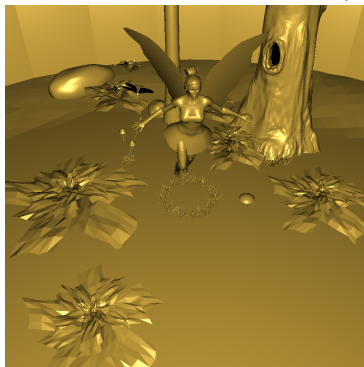
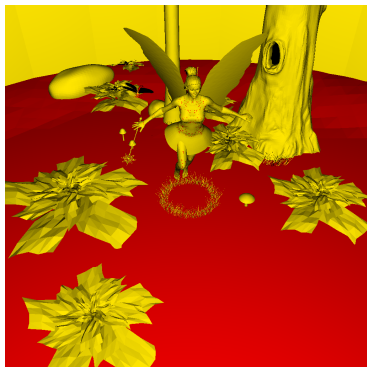
Table 3: Performance results. Includes data about total memory required without compressing scene data, the output size by compressing the scene data with gzip, the amount of the memory required to store the scene with some of our techniques enabled, the compression ratio, and finally the frame rate using any combination of the given methods. *T* triangulates the scene geometry, *P* losslessly compresses the primitive lists using arithmetic encoding, *F* losslessly compresses the faces by discarding the leading zeros of a vertex index, *V* does lossy quantization from 32-bits to 16-bits.



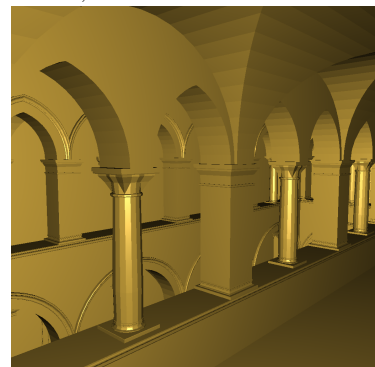
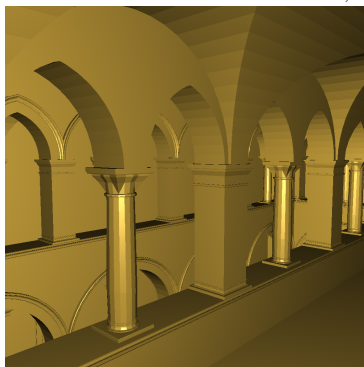
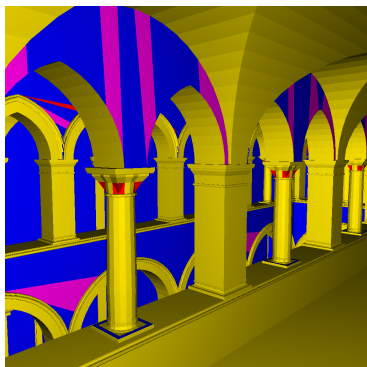
PSNR: Y: 51.72 dB, Cb: 66.10 dB, Cr: 61.41 dB



PSNR: Y: 47.84 dB, Cb: 64.43 dB, Cr: 58.51 dB



PSNR: Y: 39.22 dB, Cb: 54.87 dB, Cr: 49.03 dB



PSNR: Y: 41.23 dB, Cb: 56.71 dB, Cr: 50.69 dB

Table 4: The first column from the left allows the visualization of the primitives types in the test scenes: *triangles* are shown in red, *quads* are yellow, *triangle fans* are blue, *triangle strips* are violet. The second column shows image output without vertex compression. The third column shows image output with lossy vertex compression quantized from 32 to 16 bits.

Hough Transform-based Technique for Automated Carbon Nanocone Segmentation

Andries H. Smith[†] Jong Kwan Lee[†] Hao Hu[†] Eric S. Mandell[‡]

[†]Department of Computer Science [‡]Department of Physics & Astronomy
Bowling Green State University
Bowling Green, OH 43403, U.S.A.

{smithah, leej, haohu, meric}@bgsu.edu

ABSTRACT

A new technique to automatically segment the L-shaped carbon nanocone structures from Transmission Electronic Microscopy (TEM) images is described. The technique enables robust segmentation of the structures by exploiting a simplified Generalized Hough Transform (HT)-based processing. Exploitation of parallelism on commodity hardware is also explored for efficient processing. Effectiveness of the technique is evaluated through experiments on synthetic, simulated, and real images of carbon nanocone structures.

Keywords

Hough Transform, Carbon Nanocone Segmentation, Image Processing

1 INTRODUCTION

Automated feature segmentation has been utilized in many research and application domains (e.g., [Pie03, WCH04, YLL05, CNG09, Saf12, KIB12]) as it plays a key role in localizing objects of interest efficiently and effectively. Often, localization of objects leads to discovery of information which can help understanding of the underlying characteristics of the objects. For example, independent component analysis-based segmentation [WCH04] has been applied in medical imaging to characterize blood supply patterns which are critical for the profound analysis of cerebral hemodynamics. An automated method for localizing auroral ovals in satellite images was also introduced [CNG09]. Such a method can help scientists study the Earth's magnetic field. A fast on-line video motion segmentation method [KIB12] has also been recently presented in a multimedia application. In this paper, we introduce a new, robust automated technique for segmentation of L-shaped carbon nanocone structures from Transmission Electronic Microscopy (TEM) images.

Carbon nanocone structures represent a class of novel materials that are of high interest to investigations of carbon's role in fossil fuel, hydrogen storage, and nanotechnologies. These structures appear as two

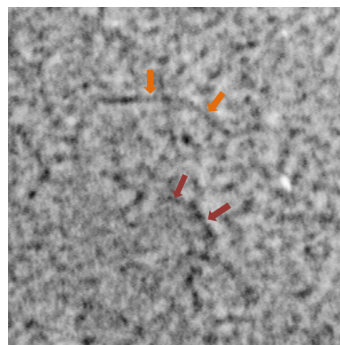


Figure 1: A sample of carbon nanocone TEM image

linear structures joined at a common point in TEM images. Moreover, the linear structures form an angle of approximately 110° or 140° for the faceted nanocones and 113° or 150° for relaxed nanocones [Man08]. (Faceted and relaxed nanocones are types of carbon nano-structures that are studied by many scientists.) A robust localization of such carbon nanocone structures is a key precursor to effective use of the TEM images in many fields, such as biosensors or nanocomputing. A sample TEM image of the carbon nanocone structures is shown in Figure 1. In the figure, two sample carbon nanocone structures (i.e., dark linear features) are marked by two sets of colored arrows for readers. As shown in the figure, other non-nanocone features are also present, the carbon nanocone structures are oriented in various ways and have varying side lengths and varying contrasts along the structure, making automated segmentation of the structures very challenging.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

The new automated segmentation of carbon nanocone structures is based on a variant of the Hough Transform (HT) [Hou62]. The new technique enables robust segmentation of the structures by utilizing a simplified binning process in the Generalized Hough Transform (GHT) [Bal81]. (GHT will be discussed in the next section.) We also exploited data parallelism on commodity hardware for efficient processing.

The paper is organized as follows. In Section 2, the related work is discussed. The new segmentation technique is introduced in Section 3. In Section 4, the experimental results and analysis are presented. Section 5 concludes this paper.

2 RELATED WORK

The related work, including the work our technique is based on, is discussed next.

2.1 Hough Transform and Its Variants

The Hough Transform (HT) [Hou62] is a well-known pattern detection method that enables recovery of global patterns in the image space via local pattern processing in the transformed parameter space. In a typical HT processing, the edge points (which present the object boundaries) in the image space are mapped to a *binned* parameter space in which dimensions of the space represent the pattern parameters. This parameter space is defined as the accumulator. After the binning process, the target pattern parameters are recovered by taking the parameters of the bins which contain the highest parameter vote counts. (While most of the HT-based methods considered the edge points in the image, direct application of HT processing on gray-scale images has also been reported [Sha96].)

The standard HT-based methods have been utilized successfully in many shape-based processing applications. However, the high memory and computational requirements of the standard HT make it difficult to apply for recovery of complex shapes. In addition, the standard HT aims to detect only analytic patterns; thus, it is difficult to apply it directly to arbitrary shapes (e.g., L-shaped carbon nanocone structures).

One way to reduce the HT's computational cost is to partition the high dimensional parameter space into lower-dimensional subspaces and then find the shape parameters in the series of lower dimensional subspaces (e.g., [HoC96]).

The Randomized Hough Transform (RHT) [XOK90] is a class of HT which can also reduce the standard HT's computational cost. While the standard HT considers all edge points, the RHT processes many sets of randomly selected edge points where each set maps to a single bin in the parameter space, allowing fewer computations during the binning process. An RHT-based method for ellipse detection was proposed by

ϕ	R
0	$\vec{r}_{0,0}, \vec{r}_{0,1}, \vec{r}_{0,2}, \dots$
$\Delta\phi$	$\vec{r}_{1,0}, \vec{r}_{1,1}, \vec{r}_{1,2}, \dots$
$2\Delta\phi$	$\vec{r}_{2,0}, \vec{r}_{2,1}, \vec{r}_{2,2}, \dots$
\vdots	\vdots

Table 1: R-table in GHT

McLaughlin [McL98]. The method benefits from the RHT's random processing and the lower dimensional sub-parameter spaces. Cao et al. [CNG09] have extended McLaughlin's RHT to employ a linear least squares fitting for a more accurate and efficient binning process.

The Generalized Hough Transform (GHT) [Bal81] is another class of HT that allows detection of both analytic and non-analytic shapes in images. Instead of binning the votes via an analytically-defined mapping scheme, the GHT utilizes a model of the object that consists of the boundary gradient direction and the distance from a reference point of the object to each point on the object boundary in a lookup table, called the *R-table*. (A reference point of the object can be any image point which can be used as an "anchor" for the object.) Specifically, the R-table defines a multi-valued mapping among the table indices corresponding to the gradient directions $\phi(\vec{p}_i)$, at the boundary points \vec{p}_i , and the vectors \vec{r} from the boundary point to the reference point \vec{c} of the object. The R-table is constructed by iterating over the object boundary points, calculating the set of \vec{r}_i for each index $\phi(\vec{p}_i)$, where i indexes a set of boundary points with the same gradient direction and \vec{r}_i is the vector from \vec{p}_i to \vec{c} . A sample format of the R-table is shown in Table 1. As shown in the table, there are multiple values $\vec{r}_{i,j}$ for each index. For each edge pixel \vec{p}_i with R-table index $\phi(\vec{p}_i)$, the potential reference points \vec{c}_i of the object is determined as $\vec{c}_i = \vec{p}_i + \vec{r}_i$. Then the binning process is performed for the corresponding values of \vec{c}_i . The GHT allows rotation-

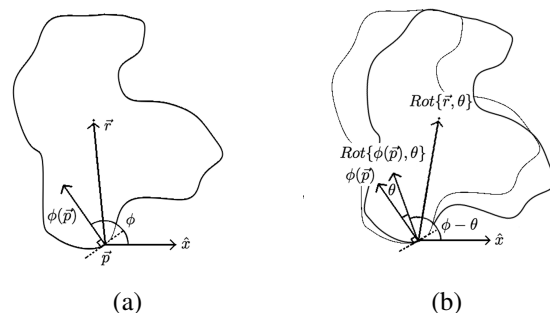


Figure 2: Illustration of the Generalized Hough Transform (GHT): (a) \vec{p}_i , $\phi(\vec{p}_i)$, and \vec{r} , (b) rotation invariant for θ by re-indexing R-table

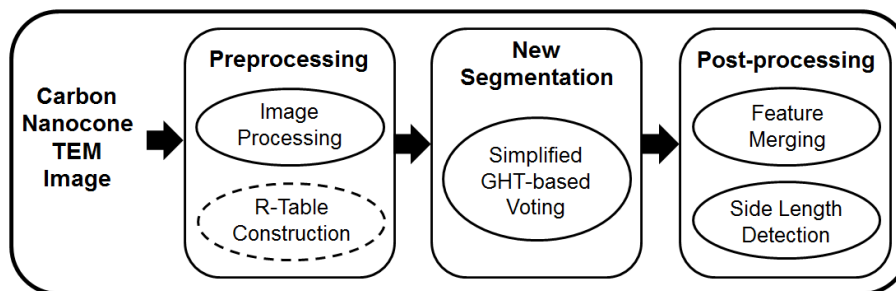


Figure 3: Overview of new GHT-based technique

invariant and scale-invariant object detection by modifying the R-table according to the rotation and scaling parameters; for scale invariant and rotation invariant detection, the R-table values are multiplied by the scaling vector or re-indexed, respectively. Figure 2 illustrates the key components of the GHT. In Figure 2 (a), the boundary point, \bar{p}_i , the gradient direction, $\phi(\bar{p}_i)$, and the vector \vec{r} are shown. Figure 2 (b) illustrates that the vector \vec{r} needs to be rotated by θ and ϕ needs to be replaced by $\phi - \theta$. This can be done by simple re-indexing of the R-table contents.

2.2 GPU-based Hough Transform

Computation using programmable graphics processing units (GPUs) has been utilized in traditional graphics as well as in an increasing number of more general-purpose applications, including simulation, data mining, analytics, databases, etc. GPU computation has also been employed in HT-based work. Here, we briefly discuss some of the key GPU-based HT work.

One early GPU-based HT was introduced by Strzodka et al. [SIM03]. In their method, programmable GPU shaders were utilized to speed up the binning process using the GPU textures as the lookup table for the shape parameter mapping. Ujaldon et al. [URG07] have presented a circle HT which uses a specific graphics pipeline (i.e., the rasterizer) for voting the circle candidates from a set of seeds computed by GPU vertex shaders. Other GPU-based HT methods include detection of more complex shapes (e.g., [ZhL05, LWN08, Deg10]). One efficient GPU HT for ellipse detection has been employed in real-time face detection [ZhL05]. Lee et al. [LWN08] presented a GPU-based HT for efficient ellipse detection in satellite imaging. Their HT uses a small set of CPU-generated random seeds in efficient GPU-based RHT processing for localization of the aurora oval.

2.3 Carbon Nanocone Segmentation

Although many methods to automatically-segment objects of interest in other types of imagery have been reported in the literature (e.g., [PRN04, LWN08]), the recently presented work by Ngatuni et al. [NLW12] is the

only automated method for carbon nanocone segmentation. Their HT-based shape detection method introduced a new parameter space for the carbon nanocone structures. Their parameter space is defined by the pre-computed distances and the orientations of a set of points on the structure. However, while the method produced accurate detection results, the binning process was applied for all combinations of the edge point pairs; thus, the method was slow. In addition, since the method weighs the pairs of the edge points that have the same distance from the point joining the two sides of the structure, the method fails to detect the structures that have different side lengths. (In Section 4, we briefly compare the method with our new technique.) The new automated segmentation technique introduced in this paper aims to overcome these weaknesses. The new technique is based on a new type of Generalized Hough Transform (GHT) which simplifies the original GHT's binning process.

3 NEW SEGMENTATION TECHNIQUE FOR CARBON NANOCONE STRUCTURES

In this section, the new technique for the automatic segmentation of the carbon nanocone structures from TEM intensity images is detailed. The key component of the technique is a simplification of GHT that enables extraction of credible L-shaped carbon nanocone structures. Exploitation of parallelism on commodity hardware is also discussed.

3.1 Overview

Figure 3 presents an overview of the technique. The technique includes preprocessing steps that “clean” the image, determine the potential carbon nanocone pixels, and compute a lookup table, the simplified GHT-based bin voting step that exploits the characteristics of L-shaped object, and post-processing steps that merge segmented structures that are very close to each other and determine the structure side lengths. (In the figure, the dotted oval for the lookup table construction step indicates that it is computed only one-time.)

3.2 Preprocessing

In the new carbon nanocone structure segmentation, a set of image processing steps are applied (1) to high-light carbon nanocone structure, (2) to remove the non-nanocone structures, and (3) to generate the edge point image for HT's binning process.

First, median filtering is applied to the intensity-inversed image to remove high spike noise and smooth out the structures. We empirically determined that 7×7 median filtering could eliminate much of the high spike noise and fill the gaps within the structures. Second, an intensity cumulating scheme is applied to the median filtered image to highlight the carbon nanocone structures. This intensity cumulating is done by replacing each pixel intensity with the maximum of the pixel intensity sums in narrowed rectangular-shaped (e.g., 15×5) image regions centered at the pixel. Since the carbon nanocone structures are linear features, this intensity cumulating step highlights (i.e., increases the contrast of) the structures and fills in the gaps within the same structures.

Third, a modified version of the Strous linear feature pixel labeling algorithm [Str00] is applied to the image "cleaned" in the first two steps. The Strous algorithm allows determination of local *bright* pixels by considering 3×3 local image region. We have modified the algorithm to consider 5×5 local image region with a less restricted constraint to label a pixel as a potential point on the carbon nanocone structures. This modification allows more points on the structures to be labeled as the potential nanocone pixels. However, this modification results in "thick" potential carbon nanocone structures. To select the points along the centerline of the structures, the morph-thinning [Dou92] is applied to the result of the modified Strous algorithm. (Morph-thinning is one of the widely-used thinning operators which employs a morphological erosion-based hit-or-miss operator to the original image with a pair of structuring elements, and then subtracts the result from the original image.)

Lastly, global and adaptive thresholdings are applied to the "cleaned" image to remove non-nanocone features. Specifically, the binary result of thresholdings is used as a mask on the morph-thinned image. For the global thresholding, its threshold is the median intensity T of the filtered image; all pixels whose intensities are less than T are considered to be non-nanocone pixels. For the adaptive thresholding [GoW02], the filtered image is sub-divided with overlapping and thresholding is applied in each sub-region. We have found empirically that sub-dividing the image into 64×64 pixel tiles with 50% overlap (e.g., the top-half of a sub-region overlaps the bottom-half of the regions that is above sub-region) produces the best results for the TEM images.

3.3 New HT for carbon nanocone segmentation

The carbon nanocone segmentation technique presented here exploits a simplified lookup table-based GHT binning scheme to recover the L-shape structure parameters.

In the technique, the coordinates of the reference point and the orientation of the L-shape are used as the shape parameters for the carbon nanocone structure. Figure 4 shows a L-shaped structure and its parameters. The reference point O is defined as the point joining the two structure edges and the orientation γ is defined as the angle between the x -axis and the vector halving the two edges. (In the figure, the side edges of the structure are denoted as the counter-clockwise and clockwise edges.)

The new segmentation technique utilizes a simplified lookup table that is designed to recover the L-shape parameters through a GHT binning process. We will call this lookup table the *L-table* in this paper. Unlike the standard GHT's lookup table, the L-table does not include many entries for different gradient directions since the gradient directions are the same for all the points on the same edge of the structure; thus, the L-table includes entries for only two gradient directions, one for each edge of the linear structure. In addition, the L-table is indexed by the distance between the reference point to the points on the edges. This indexing strategy yields a very simple scalar-valued lookup table mapping from the distance to the orientation. In addition, unlike the standard GHT, the new HT binning pro-

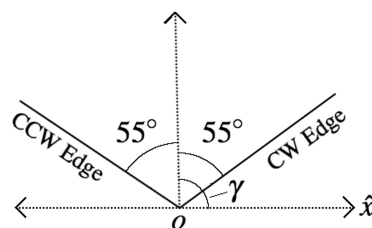


Figure 4: Illustration of a L-shape structure (representing a 110° faceted nanocone): O and γ represent the reference point and the structure orientation, respectively.

CCW edge		CW edge	
Distance	Orientation	Distance	Orientation
1	55°	1	-55°
2	55°	2	-55°
3	55°	3	-55°
\vdots	\vdots	\vdots	\vdots

Table 2: A diagrammatic representation of the L-table for 110° faceted nanocones

cess does not include a scaling of values in the L-table. Instead, a simple post-processing step to determine the edge lengths of the structure is included to reduce the computational cost. (This post-processing step is discussed in the next subsection.)

In our HT binning process, each bin of the accumulator contains two vote values; one vote counts for each edge. This two-value bin enables segmentation of L-shapes with imbalanced side lengths—even if the vote counts from one short edge is low, the vote counts from the other longer edge is high; thus, the structure parameters can be recovered.

3.4 Post-processing

It is possible for the new HT binning process to produce structures that are very close to each other. In addition, the binning process determines only the coordinates of the reference point and the orientation of the structure (but not the edge lengths). To merge structures that are close to each other into one structure and to determine the edge lengths, we post-process the output of the binning process.

To merge the structures that are very close to each other, we find the peaks in the parameter space (i.e., 3D accumulator) via a $N \times N \times N$ local search. Specifically, the merging is done by averaging the structure parameters for the local maxima above a threshold value within the local search space. We have used a threshold value that is dependent of the total number of the edge points to allow varying threshold value for different noise levels. (We empirically determined that about 3% of the total number of edge points is a good threshold value and $16 \times 16 \times 16$ -accumulator-bin is a reasonable local search space for the carbon nanocone segmentation.)

The lengths of the structure edges are determined via a simple edge extension scheme that extends the edge from the reference point along the edge directions. Each side edge is extended if there exists “enough” evidence of potential nanocone pixels within a rectangular (e.g., 7×3) region along the edge direction.

3.5 Efficient Processing

To efficiently process the new technique’s binning process, we have also explored the data parallelism on commodity hardware. While other types of parallelization are possible, we focused on multithreading in a shared memory environment. (We have also reported our preliminary results on GPU-based processing in Section 5.) Both OpenMP and Pthreads have been used in our parallel implementation. In the multithreading, a set of edge points are assigned to each thread and processed independently. The updates to the accumulator in the shared memory is done via atomic addition.

To increase the performance of the parallel implementation, we manually set the processor affinity

of each OpenMP thread using the Pthreads library. Each OpenMP thread’s processor affinity was set to a particular CPU core corresponding to its thread

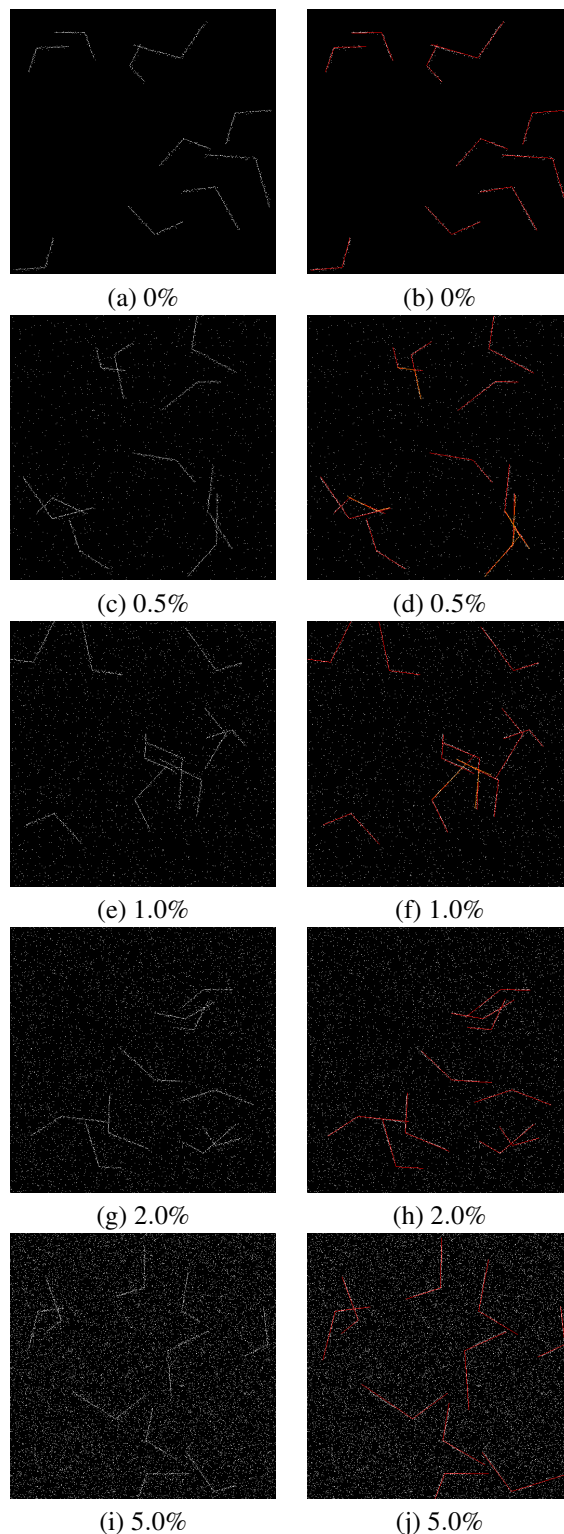


Figure 5: Results on synthetic images with perturbing noise of $\sigma = 1.5$ and five different background noise levels (0%, 0.5%, 1.0%, 2.0%, and 5.0%)

```

for  $r$  from 1 to  $360^\circ$  do
  for  $d$  from 1 to  $Max_d$  do
    for  $b_x$  from 1 to  $Width$  do
      for  $b_y$  from 1 to  $Height$  do
        for  $p$  from 1 to  $NumPatterns$  do
           $o_x, o_y = L(b_x, b_y, r, p, d)$ 
          increment  $A[p][o_x][o_y][r]$ 

```

Algorithm 1: An example of a naive implementation of the binning process

```

for  $p$  from 1 to  $NumPatterns$  do
  for  $b_x$  from 1 to  $Width$  do
    for  $b_y$  from 1 to  $Height$  do
      for  $d$  from 1 to  $Max_d$  do
        for  $r$  from 1 to  $360^\circ$  do
           $o_x, o_y = L(b_x, b_y, r, p, d)$ 
          increment  $A[p][o_x][o_y][r]$ 

```

Algorithm 2: A better implementation of the binning process with increased spatial locality of reference

ID using a `pthread_setaffinity_np()` call. We also increased locality of reference by re-ordering the algorithm. Setting processor affinity allowed for significant performance improvements for the parts of the algorithm with increased locality of reference. This can be attributed to the increased effectiveness of each of the CPU caches under these conditions.

One simple example that illustrates the consideration of locality of reference is shown in Algorithms 1 and 2. Algorithm 1 typifies an implementation of HT binning process that does not exploit the locality of reference well. Due to the row-major-order access pattern of the multidimensional accumulator, the memory access for each iteration of the nested loop results in very high number of cache misses. Algorithm 2 optimizes the naive algorithm by considering more spatial locality of reference.

4 EXPERIMENTAL RESULTS

The effectiveness of the new carbon nanocone structure segmentation technique has been benchmarked using over 1,000 synthetic images and 20 simulated and real carbon nanocone TEM images. (All of the images tested were of size 512×512 .) The benchmarking included measuring the effectiveness of the technique by considering four different errors—errors in reference point position, structure orientation and side lengths—, the false positive rate, and the match rate, and measuring the efficiency of our parallelized implementation.

We have used the Oakley cluster system of the Ohio Supercomputing Center in our experiments. The system supports 12 cores (Intel Xeon x5650 2.66 GHz CPUs) and 48 GB memory per node (up to thousands of cores in total) and 128 Nvidia Tesla M2070 GPUs in total.

Here, we note that while the cluster system provides the state-of-art computing power and our parallelized implementation is scalable to use all the system resources, our experiments were limited to amount of resources that are available on a typical PC configuration (e.g., a hyper-threaded, 6-core CPU and one programmable GPU). This limited resource usage allows to show the technique's performance for typical PC users.

For the synthetic images, we modeled the carbon nanocone structures using two straight lines joined at an angle of 110° or 140° . Modeled structures also had different random side lengths. In addition, two different types of image noise were considered; random background noise and random noise that perturb the positions of the edge points. We have considered 0%, 0.5%, 1.0%, 2.0%, and 5.0% background noise. Gaussian random noise with zero mean and σ 's of 0.0, 1.5, and 3.0 was used to perturb the edge point positions.

Figure 5 shows the five samples of the synthetic images on the left column and the segmentation results as red (and orange) overlays for the same images on the right column. These synthetic images contain five different levels of the background noise with the perturbing noise of $\sigma = 1.5$. As shown in the images, the new technique well-segmented all the structures (even for the image with the high background noise). Here, we note that the technique also produced some false positives. While some of the false positives were caused by image noise, many of them were caused by randomly oriented, nearby structures in which the edges from different structures made up a L-shape similar to a carbon nanocone. Such false positives are shown as or-

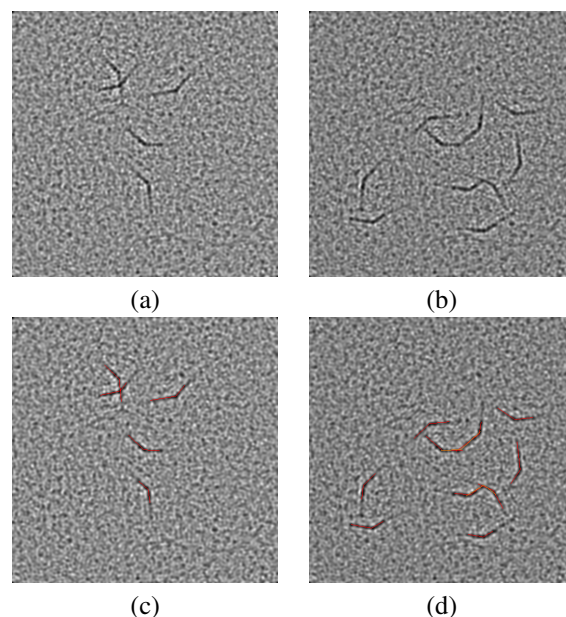


Figure 6: Results on simulated nanocone TEM images

Noise Level (# of Edge Pt.)	0.0 (704)	0.005 (2811)	0.01 (3308)	0.02 (5885)	0.05 (18845)
Ref. Pt. Position	0.51	0.41	0.51	0.47	0.67
Orientation	0.31	0.33	0.39	0.58	0.99
Side Length	1.43	1.69	2.11	3.81	16.95
Match Rate	0.99	0.99	1.00	1.00	1.00

(a) With perturbing noise of $\sigma = 0.0$

Noise Level (# of Edge Pt.)	0.0 (610)	0.005 (2683)	0.01 (3209)	0.02 (5787)	0.05 (18714)
Ref. Pt. Position	0.71	0.72	0.70	0.73	0.93
Orientation	0.52	0.55	0.63	0.83	1.19
Side Length	1.63	2.38	2.91	4.78	16.83
Match Rate	1.00	0.99	1.00	1.00	0.95

(b) With perturbing noise of $\sigma = 1.5$

Noise Level (# of Edge Pt.)	0.0 (616)	0.005 (2686)	0.01 (3220)	0.02 (5793)	0.05 (18723)
Ref. Pt. Position	1.42	1.35	1.43	1.57	1.52
Orientation	0.79	0.80	0.91	1.11	1.43
Side Length	2.98	3.37	3.45	6.02	14.95
Match Rate	0.92	0.914	0.88	0.78	0.16

(c) With perturbing noise of $\sigma = 3.0$

Table 3: Average errors on synthetic image tests: reference point position error (in pixels), orientation error (in degrees), side length error (in pixels) and average match rate

Rate	Max.	Mean	Min.	Std. Dev.
False Pos. Rate	0.22	0.14	0.00	0.05

Table 4: False positive rate on synthetic images

ange overlaps in Figure 5. For instance, all four false positives in Figure 5 (b) were from the edges of two different structures that made up a L-shape.

Tables 3 and 4 summarize the segmentation results for all the synthetic image tests. The average errors in the reference point position, orientation, and side lengths of structures, and the match rate are shown in Table 3 and four metrics (i.e., maximum, mean, minimum, and standard deviation) of the false positive errors are shown in Table 4. Table 3 also includes the number of edge points considered for different noise levels. As shown in Table 3, the new technique produced very low errors with high match rates for most of the cases. For example, synthetic image tests with a perturbing noise of $\sigma=1.5$ and background noise of 0.01% had the average errors of 0.70 in pixels, 0.63 in degrees, 2.91 in pixels for the reference point position, orientation, and side length, respectively, and had a 100% match rate. For most of the synthetic image tests, the new segmentation technique had a match rate of higher than 90%. While the average match rate for the image with the highest noise level (i.e., 5% background noise and perturbing noise of $\sigma=3.0$) was only 16%, a typical “cleaned” TEM images of carbon nanocone does not contain similar noise levels. As shown in Table 4, the technique produced about 14% false positive errors on average. However, many of the false positives were caused by the randomly oriented, nearby structures that were explained previously.

For the simulated image tests, the carbon nanocones were created using the TEM image simulation pre-

sented by Kirkland [Kir88]. (This image simulation has been used by other scientists in the field.) Figure 6 shows the segmentation results on two samples of simulated TEM carbon nanocones images. In the figure, the sub-images (a) and (b) are the simulated images with different numbers of nanocones and the sub-images (c) and (d) show the segmentation results as red (for matched structures) and orange (for false positives) overlays. As shown in the figures, the new technique segmented all of the structures for these images. Some false positives were also segmented for the simulated images. Many false positives were also caused by nearby structures in which the edges from different structures made up a L-shape similar to a carbon nanocone. For instance, Figure 6 (d) contains two of such false positives.

To analyze the effectiveness of the new segmentation technique for the real carbon nanocone TEM images, the manual segmentation results were used as the gold standard since the actual positions of the real carbon nanocone structures were unknown. For the manual segmentation, the reference point positions, the structure orientations, and the side lengths were manually-selected by a field expert. (Here, we note that the field expert indicated that manual segmentations of the structures are ambiguous and can often lead to mis-segmentations.)

Figure 7 shows the segmentation results on three samples of real images. In the figure, the sub-images (a)–(c) show the input images, the sub-images (d)–(f) show the manual segmentation results as green overlays, and the sub-images (g)–(i) show the automated segmentation results as red (for matched structures) and orange (for false positives) overlays. As shown in the figures, while the new technique segmented the carbon nanocone structures reasonably well, it also produced relatively high number of false positives (i.e., the orange overlays in the figure). Some of the false positives were from side edges from two different nearby structures making a L-shape. Others were from human bias on the subjective manual segmentation of the field expert; the field expert has agreed that some of the carbon nanocone structures were missed in the manual segmentation results after comparing the results with the automatic segmentation results. This differ-

Errors & Rates	Simulated	Real
Ref. Pt. Position	1.94	2.41
Orientation	1.14	1.95
Side Length	3.33	26.04
Match Rate	0.94	0.86
False Pos. Rate	0.05	0.59

Table 5: Average errors on simulated and real image tests: reference point position error (in pixels), orientation error (in degrees), side length error (in pixels), match rate and false positive rate

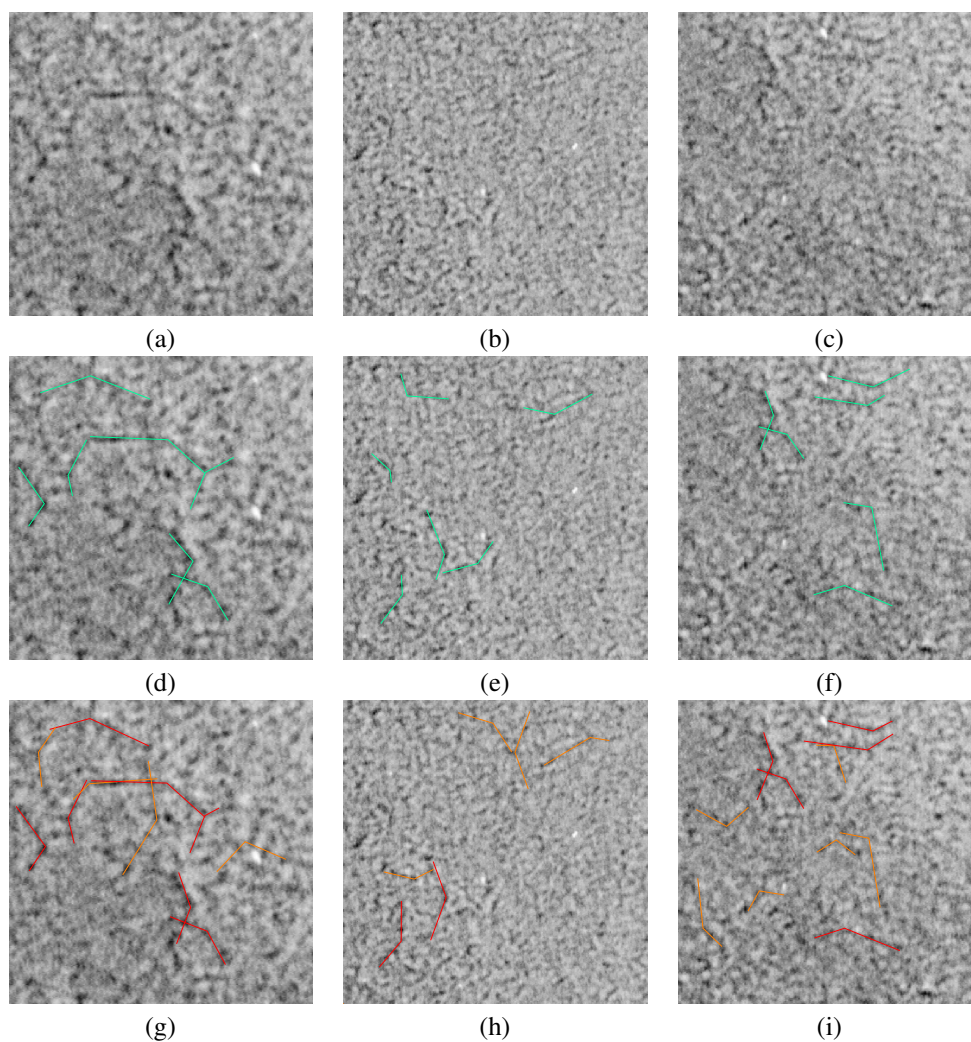


Figure 7: Results on real carbon nanocone TEM images

ence suggests that the results from the new automated segmentation technique could be the “better” segmentation results than the manually-segmented results in some cases.

The same types of errors were measured to benchmark the new technique’s effectiveness on the simulated and real images: the errors in the reference point position, orientation, side lengths, and the match rate and false positive rates. Table 5 summarizes the benchmarking. As shown in the table, the new segmentation technique produced very low errors with high match rate for the simulated images; it produced average errors of about 1.94 pixels, 1.14 degrees, 3.33 pixels for the reference point position, orientation, and side lengths, respectively, with 94% match rate and 5% false positive rate on average. The technique also produced very low average errors in the reference point position and the orientation (i.e., about 2.41 pixels and about 1.95 degrees, respectively) with a match rate of 86% for the real image testings. However, it produced somewhat a high

average side length error (i.e., about 26 pixels for both edges) and 59% false positive errors. These high side length and false positive errors were caused by the image noise on the carbon nanocone TEM images, the low-contrasted and blurry nanocone structures, and by the human bias on the gold standard (i.e., manually-segmented structures) used for the comparison.

Next, the efficiency of the new segmentation is benchmarked by measuring the execution times on our multithreading-based parallelization of the technique using the synthetic image tests. Figure 8 summarizes the efficiency benchmarking results. In the figure, the speedups of the multithreading using 1, 2, 4, 6, 8, 10, and 12 threads are shown for the images with different noise levels. For each noise level, the total number of edge points is also noted in parentheses in the figure. (We note again that only up to 12 threads were used in our experiments for the reason discussed previously. However, our parallelization is scalable to use more threads to achieve real time processing.) As shown

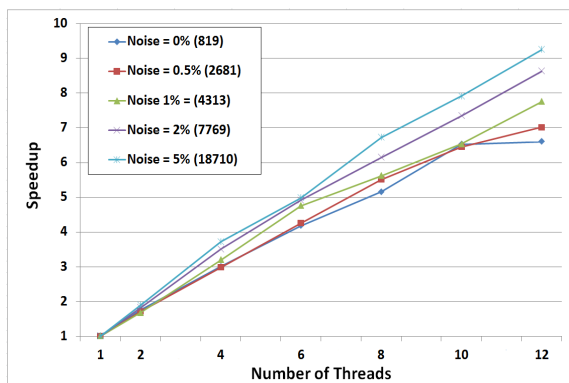


Figure 8: Speedups of multithreading-based parallel processing on synthetic images with different background noise levels with perturbing noise of $\sigma = 1.5$, (the total # of edge points are shown in parenthesis)

in the figures, the multithreaded version performed and scaled reasonably well. For example, when using 12 threads, we have achieved 6.59 (i.e., from 34.48 seconds to 5.23 seconds) and 9.24 (i.e., from 187.60 seconds to 20.31 seconds) speedups for noise levels of 0.0% and 0.05%, respectively over the single threaded version. The execution times (shown in Table 6) varied for different noise levels by about 18% to 23% since the total number of edge points considered in the images varied about from 819 to 18,710 edge points on average. Here, we also note that the performance improvements discussed in Section 3.5 contributed about 10% performance increase.

Lastly, a brief comparison of our automated segmentation and the only prior carbon nanocone segmentation method by Ngatuni et al. [NLW12] is reported. The comparison experiments included measuring the same types of the errors and the execution times using synthetic images with different types of L-shaped linear structures and using one real image (shown in Figure 7 (a)). While our technique performed well on the synthetic image (i.e., errors, match rate, and false positive rate similar to the ones reported in Table 3), their method performed poorly for the synthetic images with L-shapes that have different side lengths since their method weighs the pairs of edge points that have the same distance from the reference point in their binning process. Their method also segmented only one structure for the real image. In addition, their method ran about 3–10 times slower than the serial version of the new segmentation technique.

5 CONCLUSION

We have presented and evaluated a new technique of segmenting L-shaped structures in high-resolution carbon nanocone TEM images. The new technique involves use of a simple lookup table that enables efficient

Noise Level (# of Edge Pt.)	0.0 (819)	0.005 (2681)	0.01 (4313)	0.02 (7769)	0.05 (18710)
1 thread	34.48	47.56	62.23	93.67	187.60
2 threads	19.60	27.62	37.10	51.39	98.81
4 threads	11.46	15.94	19.48	26.70	50.43
6 threads	8.26	11.16	13.08	19.07	37.65
8 threads	6.69	8.63	11.08	15.24	27.94
10 threads	5.29	7.37	9.53	12.77	23.73
12 threads	5.23	6.78	8.03	10.87	20.31

Table 6: Execution times (in seconds) of multithreading-based parallel processing on synthetic images with different background noise levels with perturbing noise of $\sigma = 1.5$, (the total # of edge points are shown in parenthesis)

GHT binning process. The new automated segmentation technique produces reasonably good results for a variety of carbon nanocone images and provides a consistent solution. In addition, the multithreading-based parallelization of the technique with the exploitation of locality of reference performs the computationally expensive processing of the segmentation efficiently.

For the future work, other preprocessing steps to distinguish the carbon nanocone structures from other features in the image can be explored. Improvement of the load balancing in the multithreading may also be possible. We plan to achieve real-time processing of the technique by considering a fine-grained parallel approach using programmable GPU processing (e.g., using CUDA) and by considering MPI-based multiprocessing. Our preliminary results on CUDA-based GPU processing shows about 4–6 times performance speedups. However, the limit on the GPU’s shared memory and the inefficient atomic operation on the accumulator update make the GPU-based algorithm challenging. One challenge of the MPI-based multiprocessing is the overhead to gather the large accumulator. For example, the MPI_Allreduce operation for gathering the 3D accumulator took about 2.2 seconds in a 12-node configuration. One simple solution to reduce this overhead could be to increase the size of each bin in the accumulator (with a cost of relatively small accuracy).

6 ACKNOWLEDGEMENTS

This work was supported in part by an allocation of computing time from the Ohio Supercomputer Center. We also appreciate Ronald Ngatuni for providing assist on running experiments using their method [NLW12].

7 REFERENCES

- [Bal81] Ballard, D., Generalized Hough Transform to Detect Arbitrary Patterns, *IEEE Trans. of Pattern Analysis and Machine Intelligence*, Vol. 13 (2), 1981, pp. 111–122.
- [CNG09] Cao, C., Newman, T.S., and Germany, G.A., New Shape-based Auroral Oval Segmentation Driven by LLS-RHT, *Pattern Recognition*, Vol. 42 (5), 2009, pp. 607–618.

- [Deg10] Degirmenci, M., Complex Geometric Primitive Extraction on Graphics Processing Unit, *Journal of WSCG*, Vol. 18, 2010, pp. 129–134.
- [Dou92] Dougherty, E.R., *Introduction to Morphological Image Processing*, SPIE-International Society for Optical Engine, 1992.
- [GoW02] Gonzalez, R.C. and Woods, R.E., *Digital Image Processing*, second ed., Prentice Hall, 2002.
- [HoC96] Ho, C., and Chen, L., A High-speed Algorithm for Elliptical Object Detection, *IEEE Transaction on Image Processing*, Vol. 5 (3), 1996, pp. 547–550.
- [Hou62] Hough, P.V.C., Method and Means for Recognizing Complex Patterns, U.S. Patent, 3,069,654, 1962.
- [Kir88] Kirkland, E.J., *Advanced Computing in Electron Microscopy*, Plenum Press, 1988.
- [KIB12] Klicnar, L., and Beran, V., Robust Motion Segmentation for On-line Application, in Proc., 20th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'12), Plzen, Czech Republic, 2012, pp. 205–212.
- [KXO90] Kultanen, P., Xu, L., and Oja, E., Randomized Hough Transform (RHT), in Proc., 10th Int'l Conf. on Pattern Recognition, Atlantic City, 1990, pp. 631–635.
- [LWN08] Lee, J.K., Wood, B.A., and Newman, T.S., Very Fast Ellipse Detection using GPU-based RHT, in Proc., 19th Int'l Conf. on Pattern Recognition, Tampa, 2008, pp. 1–4.
- [Man08] Mandell, E.S., *Electron Beam Characterization of Carbon Nanostructures*, Ph. D. Dissertation at the University of Missouri-Rolla, 2008.
- [McL98] McLaughlin, R.A., Randomized Hough Transform: Improved Ellipse Detection with Comparison, *Pattern Recognition Letters*, Vol. 19 (3–4), 1998, pp. 299–305.
- [NLW12] Ngatuni, R., Lee, J.K., West, L., and Mandell, E.S., New Hough Transform-based Algorithm for Detecting L-shaped Linear Structures, in Proc., 2012 Int'l Conf. on Image Processing, Computer Vision, and Pattern Recognition (IPC'12), Las Vegas, 2012, pp. 641–646.
- [Pie03] Pietrowcew, A., Face Detection in Colour Images using Fuzzy Hough Transform, *Opto-Electronics Review*, Vol. 11 (3), 2003, pp. 247–251.
- [PRN04] Pintaviroj, C., Romputtal, A., Ngamlamiad, A., Withayachumnankul, W., and Hamamoto, K., Ultrasonic Refractive Index Tomography, *Journal of WSCG*, Vol. 12, 2004, pp. 333–339.
- [Saf12] Safdar, K., Detecting and Removing Islands in Graphics-Rendering-Based Computations of Lower Envelopes of Plane Slabs, in Proc., 20th Int'l Conf. on Central Europe on Computer Graphics, Visualization, and Computer Vision (WSCG'12), Plzen, Czech Republic, 2012, (abstract).
- [Sha96] Shapiro, V.A., On the Hough Transform of Multi-level Pictures, *Pattern Recognition*, Vol. 29 (4), 1996, pp. 589–602.
- [SIM03] Strzodka, R., Ihrke, I., and Magnor, M., A Graphics Hardware Implementation of the Generalized Hough for Fast Object Recognition, Scale, and 3D Pose Detection, in Proc., 12th Int'l Conf. on Image Analysis and Processing '03, Mantova, Italy, 2003, pp. 188–193.
- [Str00] Strous, L.H., Loop Detection, http://www.lmsal.com/~schwand/stereo/2000_-easton/cdaw.html, 2000.
- [URG07] Ujaldon, M., Ruiz, A., and Guil, N., On the Computation of the Circle Hough Transform by a GPU Rasterizer, *Pattern Recognition Letters*, Vol. 29 (3), 2007, pp. 309–318.
- [WCH04] Wu, Y.-T., Chen, H.-Y., Hung, C.-I., Kao, Y.-H., Guo, W.-Y., Yeh, T.-C., and Hsieh, J.-C., Segmentation of Hemodynamics from Dynamic-Susceptibility-Contrast Magnetic Resonance Brain Image Using Sequential Independent Component Analysis, in Proc., 12th Int'l Conf. on Central Europe on Computer Graphics, Visualization, and Computer Vision (WSCG'04), Plzen, Czech Republic, 2004, pp. 267–274.
- [XOK90] Xu, L., Oja, E., and Kultanen, P., A New Curve Detection Method: Randomized Hough Transform (RHT), *Pattern Recognition Letters*, Vol. 11 (5), 1990, pp. 331–338.
- [YLL05] Yu, X., Lai, H.C., Liu, S.X.F., and Leong, H.W., A Gridding Hough Transform for Detecting the Straight Lines in Sports Video, in Proc., IEEE Int'l Conf. on Multimedia and Expo, Amsterdam, The Netherlands, 2005, pp. 1–4.
- [ZhL05] Zhang, S., and Liu, Z., A Robust, Real-Time Ellipse Detector, *Pattern Recognition*, Vol. 38 (2), 2005, pp. 273–287.
- [ZhS84] Zhang, T.Y., and Suen, C.Y., A Fast Parallel Algorithm for Thinning Digital Patterns, *Communications of the ACM*, Vol. 27 (3), 1984, pp. 236–239.

Joining Meshes with a Local Approximation and a Wavelet Transform

Anh-Cang PHAN
Aix-Marseille University
CNRS, LSIS UMR 7296
13009, Marseille, France
Anh-cang.Phan@univ-amu.fr

Romain RAFFIN
Aix-Marseille University
CNRS, LSIS UMR 7296
13009, Marseille, France
Romain.Raffin@univ-amu.fr

Marc DANIEL
Aix-Marseille University
CNRS, LSIS UMR 7296
13009, Marseille, France
Marc.Daniel@univ-amu.fr

ABSTRACT

Constructing a smooth surface of a 3D object is an important problem in many graphics applications. Subdivision methods permit to pass easily from a discrete mesh to a continuous surface. A generic problem arising from subdividing two meshes initially connected along a common boundary is the occurrence of cracks or holes between them. Therefore, we propose a new method for joining two meshes with different resolutions using a tangent plane local approximation and a Lifted B-spline wavelet transform. This method generates a connecting mesh where continuity is controlled from one boundary to the other. This guarantees that the discrete continuity between these mesh areas is preserved and the connecting mesh can change gradually in resolution between coarse and fine areas.

Keywords

Mesh connection, smoothness, local approximation, B-spline wavelets.

1 INTRODUCTION

3D object models with complex shapes are generated by a set of assembled patches or separate mesh areas which may be at different resolution levels, even with different subdivision schemes. Cracks, gaps or holes may appear on their surfaces because of having a difference in subdivision schemes or a difference in resolution levels between adjacent faces.

In order to overcome these drawbacks and particularly cracks, this paper presents a new technique creating a smooth connecting surface linking two meshes with different resolutions and with different subdivision schemes. We aim at constructing a high quality connecting mesh between two selected areas of a model so that we can preserve the “continuity” between these selected mesh areas to generate a smooth surface. It means that the curvatures must be “continuous” on the boundaries, which must be studied in terms of discrete curvatures, the latter being not presented here. The discrete boundary curves of the connecting mesh are created by a linear interpolation, a tangent plane local approximation, and a Lifted B-spline wavelet transform. They respect the local curvatures and change their point

densities gradually from coarse to fine and conversely. Our contributions are as follows: 1) Provide a mesh joining solution by constructing a high quality connecting mesh (CM) between two meshes defined with subdivision surfaces, each mesh being at a different subdivision level; 2) Detect and remove cracks on the surface caused by a difference in resolution between neighboring faces; 3) Apply a local tangent plane reconstruction and a wavelet transform to position newly inserted vertices on the expected surface. This enables us to reconstruct smooth connecting surfaces from boundary vertices of the two meshes. Therefore, the continuity between the meshes will be preserved; 4) Allow filling holes, pasting meshes, and joining 3D objects to generate a smooth discrete surface with a natural shape and visually fair connectivity.

The remaining of the paper is organized as follows: Section 2 and 3 detail the previous and related works for mesh connection. Our algorithm is described in section 4 and details are given in section 5. We show and compare results of our algorithm in section 6. Finally, we draw the conclusion in section 7.

2 PREVIOUS WORKS

Subdivision surfaces have been used widely in fields of geometric modeling, computer graphics for shape design, animation, multi-resolution modeling or even as movie production, game engines and many other engineering applications. Today one can find a variety of subdivision schemes for geometric design and graphical applications such as Catmull-Clark [Cat98], Doo-Sabin [Doo78], Butterfly [Dyn90], Loop [Loo87], etc.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

In addition, the theory of wavelets has been applied successfully in the areas of computer graphics as surface reconstruction, subdivision, multi-resolution analysis, etc. Many subdivision methods applying wavelet-based multiresolution analysis of an arbitrary surface were introduced in [Sto96, Lou97, Mal98, Zor00], etc. Moreover, subdivision wavelets with the lifting scheme have been developed in [Swe98, Ber04a]. This latter could be an interesting approach for CAD applications, like surface reconstruction.

Commonly, a subdivision of the entire input mesh is not necessary nor desired while one only needs to subdivide some areas of the mesh to add details on the object or obtain a smoother surface. This is important to reduce the number of faces of the mesh as well as the unnecessary subdivision levels, and save the refinement time or the storage space. Some research [Hus10, Hus11, Pak07] are related to the incremental subdivision method with Butterfly, Loop and Catmull-Clark schemes. The main goal of these methods is to generate a smooth surface by refining only some selected areas of a mesh and remove cracks by simple triangulation. However, this simple triangulation has some undesired side-effects. It changes the connectivity, the valence of odd vertices, and produces high valence vertices leading to long faces. This not only alters the limit subdivision surface, but also reduces its smoothness. It creates ripple effects on the subdivision surface. Moreover, these methods recommend adaptively refining the areas of interest using the same subdivision scheme. A surrounding part of the area outside the adaptively subdivided area is also refined in the most common case in order to reduce brutal normal deviation. The error is computed at each step and the subdivision is stopped once this error reaches a user specified threshold. Users need to compute and control over a number of subdivision steps.

On the other hand, such models of 3D objects are formed by assembled patches or meshes. Thus, available methods to join these meshes along boundary curves between them are of immediate practical interest. The main challenge in designing a mesh connection algorithm is to guarantee that the continuity between the meshes is preserved, while the resolution between them is changed gradually. In addition, the algorithm would be simple, and efficient. This problem has been studied extensively and there are several considerable research works relevant to connecting or joining meshes in [Bar95, Fu04]. These methods consist in connecting the meshes of a surface at the same resolution level which adopt various criteria to compute the planar shape from a 3D surface patch by minimizing their differences. They are computationally expensive and memory consuming. Recently, Di Jiang and F. Stewart [Jia07] introduced the joining algorithms based on the use (as a supplement to absolute error

criteria) of normal-vector error criteria for the discrepancy between the surface patch and the associated mesh patch. They join given meshes together while maintaining a proxy for the normal-vector error, as well as the absolute error. However, these algorithms adjust the vertices of the input meshes in a way that constrains them to lie in a transfinite interpolation defined by Whitney extension. Therefore, they can produce large changes in the normal direction of triangles near the mesh boundary, even turn the triangles upside down by the joining process. Moreover, the algorithms do not mention the continuity and the progressive change in resolution between meshes after joining them together.

In [Phan12], a mesh connection method based on a RBF local interpolation and a wavelet transform has been introduced. In this paper we propose a more reliable method for mesh connection which allows us to construct a high quality connecting mesh and a continuous surface. The goal is to gain in both time of computation and surface quality. This new method is based on a tangent plane local approximation and a wavelet transform without solving any linear system, handling cracks, modifying the original boundaries of mesh areas of a model and the closest faces around the boundaries during the connecting process. We will compare both methods in section 6.

3 BACKGROUND

3.1 Wavelet-based Multiresolution representation of curves and surfaces

Wavelet is receiving a lot of attention due to the practical interest of 3D modeling in a large range of applications, such as Computer Graphics and CAD [Mal98, Ols08]. Wavelet tool can be used to derive a hierarchical multi-resolution representation of curves and surfaces [Lou97]; a smooth curve at different resolution levels [Ols08]; an overall shape edition of a curve while preserving its details [Suc09]; and a curve approximation [Kho00]. Wavelet analysis provides a set of tools to represent functions hierarchically [Sto96]. The coarse scaling function represents coarse curves or surfaces and encodes an approximation of the function, while the wavelet function represents the difference between coarse and fine curves or surfaces, and encodes the missing details. Many subdivision wavelets have been proposed in [Ber02, Ber04a, Sam04]. They allow a decomposition of curves and surfaces at different resolutions while maintaining geometric details. In addition, the combination of B-splines and wavelets leads to the idea of B-spline wavelets [Ber04a]. B-spline wavelets form a hierarchical basis for the space of B-spline curves and surfaces in which every object has a unique representation. Taking advantage of the lifting scheme [Swe96b], the Lifted B-spline wavelet [Ber02] is a fast computational tool for multiresolution analysis

of a given B-spline curve with a computational complexity linear in the number of control points. They allow representing B-spline curves at multiple resolution levels, editing curves, etc.

The Lifted B-spline wavelet transform includes the forward and backward B-spline wavelet transform. From a fine curve at the resolution level J , C^J , the forward B-spline wavelet transform decomposes C^J into a coarser approximation of the curve, C^{J-1} , and detail (error) vectors. The detail vectors are a set of wavelet coefficients containing the geometric differences with respect to the finer levels. The backward B-spline wavelet transform can be used to reconstruct fine curves from a coarse curve and detail vectors. Given a coarse curve at the resolution level $J-1$, C^{J-1} , the backward B-spline wavelet transform synthesizes C^{J-1} and the detail vectors into a finer curve, C^J .

In our approach, we apply the Lifted B-spline wavelet transform for multiresolution analysis of discrete boundary curves of a connecting mesh.

3.2 Tangent Plane Local Approximation for implicit surface reconstruction

Reconstruction of 3D surfaces from point samples is a well-studied problem in computer graphics. It allows fitting of scanned data, filling of surface holes, connecting and remeshing of 3D complex models. Implicit surface methods can directly reconstruct approximating surfaces from 3D scattered data set, such as moving least squares (MLS) [Lev03], implicit surface methods [Car01, Cas05], etc. We can classify the methods as either global or local approaches. Global fitting methods use the whole input data to compute implicit functions. Their disadvantage is that the computational complexity rapidly increases consequently to the data set size. Moreover, they present the well-known feats to discard local details (which can be an advantage or also a disadvantage). Therefore, it is difficult to use these methods directly to reconstruct implicit surfaces from large point sets consisting of more than several thousands of input points. Practical solutions on large point sets involve the local fitting methods to reconstruct the surfaces such as RBF local interpolations [Bra06, Cas05], adaptive RBF reduction and fast RBF methods [Car01]. Both methods require the construction of linear constraints on the control points for each interpolation point and thus the definition of a system of linear equations. The surface reconstruction can be obtained by solving this system. However, without adding off-surface constraints, the linear system may become trivial and we cannot solve it to specify implicit function values.

In order to extrapolate local frames (tangents, curvatures) between two meshes, we need a local approximation method on the points that will be projected

[Hop92, Ale04, Lev03]. We choose a tangent plane based local method to approximate the expected surface using a set of local tangent planes computed at each sample point (see in Fig. 1b) because this method does not require such information as methods depending on off-surface constraints. This method enables us to reconstruct smooth implicit surfaces from a set of control points.

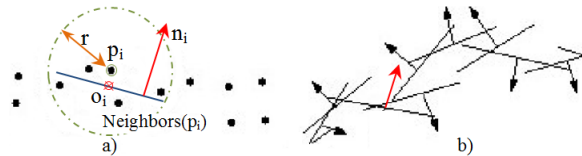


Figure 1: Nearest neighbors for tangent plane estimation.

It first considers subsets of nearest neighboring points to estimate local tangent planes as shown in Fig. 1a, and then defines an implicit function as a signed distance to the tangent plane for the data points. For example, Hoppe's method [Hop92] reconstructs a surface from a set of all unorganized points scattered on or near the surface using a contouring algorithm and the parameters k, ρ, δ defined by the user, where k is the number of nearest neighbors, ρ and δ are the thresholds of the density and noise. Thus, the considered data set is large and can contain noise. Our approach is inspired from this method. However, since our model is already a mesh model and not a set of sparse data points, we can completely determine nearest neighbors based on the connecting edges without using the parameters k, ρ , and δ . Additionally, the connecting mesh is constructed from boundary vertices of meshes and their neighbors. Thus, the cardinality of the data set is reduced. We are also able to compute the approximation error (RMS error).

Given a set of data points $P = \{p_i\} \in \mathbb{R}^3$ of a surface CM, we would like to find a signed distance function $f(p)$ from an arbitrary point $p \in \mathbb{R}^3$ to the surface CM. However, because CM is an unknown surface, the authors approximate the surface using a set of tangent planes computed at each data point as shown in Fig. 1b. The tangent plane and the signed distance function are computed as described in the following section.

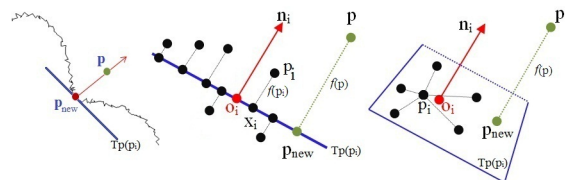


Figure 2: Estimation of a tangent plane and the projection of an arbitrary point onto it.

3.2.1 Estimation of a tangent plane

Let $Tp(p_i)$ be the tangent plane corresponding to point p_i and passing through a centroid point o_i . An arbitrary

point p is projected onto tangent plane $Tp(p_i)$ which has point o_i closest to point p . A point p_{new} is an orthogonal projection of point p onto $Tp(p_i)$ as illustrated in Fig. 2.

Tangent plane $Tp(p_i)$ is determined by passing through point o_i with unit surface normal n_i as follows:

- **Find local neighbors of each data point:** For each point $p_i \in \mathbb{R}^3$, we find a set of nearest neighbors of p_i denoted $Neighbors(p_i)$.
- **Compute a centroid point on a tangent plane:** For each point $p_i \in \mathbb{R}^3$, we compute the centroid point o_i based on all nearest neighbors of p_i :

$$o_i = \frac{\sum_{p_j \in Neighbors(p_i)} p_j}{N} \quad (1)$$

Where N is the number of the neighbors of p_i .

- **Estimate a normal vector of a tangent plane:** The principal component analysis (PCA) method is used to estimate normal n_i of $Tp(p_i)$. The point covariance matrix $CV_i \in \mathbb{R}^{3 \times 3}$ from the neighbors of p_i is first computed:

$$CV_i = \sum_{p_j \in Neighbors(p_i)} (p_j - o_i)^T (p_j - o_i) \quad (2)$$

We then compute eigenvalues $\lambda_{i,1} \geq \lambda_{i,2} \geq \lambda_{i,3}$ of CV_i associated with unit eigenvectors $v_{i,1}, v_{i,2}, v_{i,3}$. Since normal n_i is the eigenvector corresponding to the smallest eigenvalue, we choose to be either $v_{i,3}$ or $-v_{i,3}$. The choice determines the tangent plane orientation [Hop92].

3.2.2 Construction of a signed distance function

Our goal is to find a signed distance function $f(p)$ from an arbitrary $p \in \mathbb{R}^3$ to CM. The function $f(p)$ is the distance between p and the closest point $p_{new} \in CM$. Since CM is the unknown surface, we find a tangent plane $Tp(p_i)$ which is a local linear approximation of CM and passes through the centroid o_i closest to p . Therefore, the signed distance $f(p)$ to CM is represented as the signed distance $dist(p, p_{new})$ between p and its projection p_{new} onto $Tp(p_i)$ (see in Fig. 2). The function $f(p)$ satisfies the local approximation constraint defined by:

$$f(p) = dist(p, p_{new}) = (p - o_i) \cdot n_i \quad (3)$$

and p_{new} by:

$$p_{new} = p - (f(p) \cdot n_i) \quad (4)$$

3.2.3 Evaluation of the error of the tangent plane based local approximation

The local constraint (3) is too strict. Thus, if the data are noisy, the accuracy of the surface reconstruction is evaluated by the error of the approximation defined by equation (3). Since the approximation is based on fitting a plane to a set of local neighboring points, the minimize least squares (MLS) error [Ale03, Dor97, Lev03] is commonly used to evaluate the local approximation error. The MLS error evaluated at $p_i \in P$ is calculated as the sum of the squared distances from the local neighbors of point p_i to $Tp(p_i)$:

$$E_{MLS}(p_i) = \sum_{p_j \in Neighbors(p_i)} ((p_j - o_i) \cdot n_i)^2 \quad (5)$$

Additionally, we can use the root mean square (RMS) error [Sar11] to evaluate the local approximation error. The RMS error evaluated at p_i is:

$$E_{RMS}(p_i) = \sqrt{\frac{\sum_{p_j \in Neighbors(p_i)} ((p_j - o_i) \cdot n_i)^2}{N_k}} \quad (6)$$

4 METHOD OVERVIEW

4.1 Notation

In order to lighten notations, we decided not to use vectorial notations for all the notations or equations having vectorial relations. Moreover, we denote the position vector \vec{Op} of a vertex p by p , where O is the frame origin. Each multiplication of a scalar value and a vector is understood as the vector components multiplied by the scalar value.

Let M_1 and M_2 be two meshes subdivided at different resolution levels, and p_i, q_k their vertices. An edge connecting p_i to q_k is denoted e_i or $p_i q_k$. An edge is usually shared by two faces. If it is shared by only one, it corresponds to a boundary edge and its end vertices are called boundary vertices. We need to construct a connecting mesh CM between meshes M_1 and M_2 so that the continuity between them can be preserved as illustrated in Fig. 3. First we will introduce the notations relevant to the algorithm:

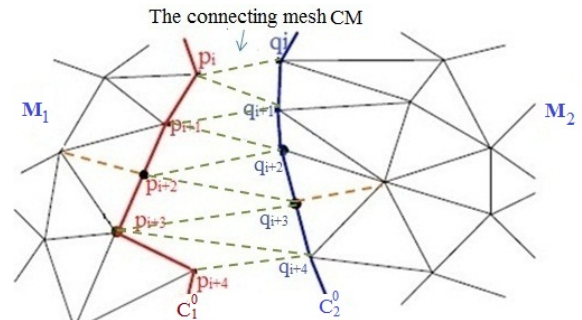


Figure 3: Topology representation of the algorithm.

- s : number of intermediate discrete curves (also called the number of newly created boundary curves) of CM created between M_1 and M_2 (see Fig. 4). It is a user parameter computed based on the distance between two original boundaries of M_1 and M_2 and it controls the resolution of CM.
- j : order number of the decomposition step to create intermediate discrete curves, also called the level. Since two boundary curves between M_1 and M_2 will be created at each level j , j is in $[1, \frac{s}{2}]$.
- C_1^j and C_2^j : two boundary curves of CM at level j . C_1^0 and C_2^0 are the two original boundary curves of meshes M_1 and M_2 .
- $N(C_1^j)$: number of vertices of boundary curve C_1^j at level j . It corresponds to the density of vertices of boundary curve C_1^j .
- p_i^j, q_i^j : vertices i on boundary curves C_1^j and C_2^j . ($p_i^0 = p_i$ and $q_i^0 = q_i$)
- L_1^j : list of the boundary vertex pairs (p_i^{j-1}, q_k^{j-1}) .
- L_2^j : list of the boundary vertex pairs (q_k^{j-1}, p_i^{j-1}) .

4.2 CM2D-TPW algorithm

The idea is to create new boundary curves C_1^j and C_2^j between M_1 and M_2 based on the previously created boundary curves C_1^{j-1} and C_2^{j-1} using the tangent plane local approximation and the Lifted B-spline wavelet transform. After that, we connect each new boundary curve C_1^j to C_1^{j-1} , and C_2^j to C_2^{j-1} . C_1^j is created in a direction from C_1^{j-1} to C_2^{j-1} and conversely for C_2^j . Therefore, **this algorithm is called the algorithm of connecting mesh in two directions based on the tangent plane local approximation and the Wavelet transform** (CM2D-TPW). The algorithm consists of the following main steps detailed in the next sections:

- **Step 1.** Boundary detection: read the input geometry model of two meshes M_1 and M_2 . Detect and mark boundary vertices of the two boundaries C_1^0 and C_2^0 in M_1 and M_2 .
- **Step 2.** Boundary vertex pairs and boundary curve creation: for each level j , we pair the boundary vertices of C_1^{j-1} and C_2^{j-1} based on the distance between them. If this distance is too narrow (smaller than a certain threshold), we go to Step 3 to connect the boundary curve pair (C_1^{j-1}, C_2^{j-1}) . In contrast, we create two new boundary curves C_1^j, C_2^j . The boundary curve creation first produces vertices of two new boundary curves from the paired boundary vertices by a linear interpolation, and then projects

them onto the expected surface CM using a tangent plane local approximation. It finally refines or coarsens these new boundary curves by applying wavelet transforms and vertex insertion and deletion operations.

- **Step 3.** Boundary curve connection: perform a boundary triangulation for each boundary curve pair (C_1^{j-1}, C_1^j) and (C_2^{j-1}, C_2^j) .
- **Step 4.** Repeat steps 2 and 3 until both mesh areas M_1 and M_2 have been connected or patched by all newly created triangles.

5 MESH CONNECTION

5.1 Boundary vertex pairs

In order to create boundary curves between two meshes M_1 and M_2 by interpolating previously created boundary curves, we pair the boundary vertices $p_i^{j-1} \in C_1^{j-1}$ with $q_k^{j-1} \in C_2^{j-1}$ and vice versa based on the distances between them. Since the densities of vertices of both boundary curves are different, we need to create two lists of the closest boundary vertex pairs L_1^j and L_2^j .

Assume that j is the current level, for each boundary vertex $p_i^{j-1} \in C_1^{j-1}$, we search for and insert into L_1^j the corresponding paired vertex $q_k^{j-1} \in C_2^{j-1}$ such that: $(\forall q \in C_2^{j-1}, dist(p_i^{j-1}, q_k^{j-1}) \leq dist(p_i^{j-1}, q))$, where the notation $dist(p_i^{j-1}, q_k^{j-1}) = \|p_i^{j-1} - q_k^{j-1}\|$ is the Euclidean distance between p_i^{j-1} and q_k^{j-1} . The list of boundary vertex pairs L_2^j is created similarly.

5.2 Boundary curve creation

The basic idea is to create two new boundary curves C_1^j and C_2^j from the paired vertices at each level j . Paired vertices are obtained by the shortest distances between vertices of each boundary. New boundary vertices $p_i^j \in C_1^j$ and $q_k^j \in C_2^j$ are created by boundary vertex pairs $(p_i^{j-1}, q_k^{j-1}) \in L_1^j$ and $(q_k^{j-1}, p_i^{j-1}) \in L_2^j$ respectively. A new boundary curve C_1^j is created in a direction from C_1^{j-1} to C_2^{j-1} and a new boundary curve C_2^j is created in a direction from C_2^{j-1} to C_1^{j-1} as shown in Fig. 4.

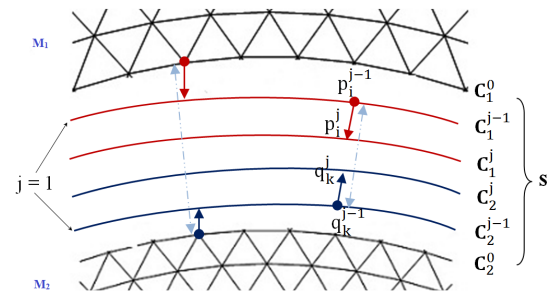


Figure 4: Boundary curves created in two directions.

We assume $N(C_1^0) \leq N(C_2^0)$ and let the density of vertices of the two boundary curves C_1^j and C_2^j be two functions $N(C_1^j)$ and $N(C_2^j)$ defined by:

$$\begin{aligned} N(C_1^j) &= N(C_1^0) + \frac{j}{s+1} [N(C_2^0) - N(C_1^0)] \\ N(C_2^j) &= N(C_2^0) - \frac{j}{s+1} [N(C_2^0) - N(C_1^0)] \end{aligned} \quad (7)$$

The boundary curves are created in three phases.

5.2.1 Phase 1: Create vertices of two new boundary curves by a linear interpolation.

- Create vertices of the discrete boundary curve C_1^j in a direction from C_1^{j-1} to C_2^{j-1} (see Fig. 4): for each boundary vertex pair $(p_i^{j-1}, q_k^{j-1}) \in L_1^j$, we apply the linear interpolation equation (8) to create new boundary vertices $p_i^j \in C_1^j$.

$$p_i^j = p_i^{j-1} + \frac{j}{s+1} (q_k^{j-1} - p_i^{j-1}) \quad (8)$$

Where i are the subscripts of boundary vertices of C_1^j , $1 \leq i \leq N(C_1^{j-1})$, and k are the subscripts of boundary vertices of C_2^{j-1} , $1 \leq k \leq N(C_2^{j-1})$.

- In the same way, we create the new boundary vertices $q_k^j \in C_2^j$ by (9).

$$q_k^j = q_k^{j-1} + \frac{j}{s+1} (p_i^{j-1} - q_k^{j-1}) \quad (9)$$

Where k are the subscripts of boundary vertices of C_2^j , $1 \leq k \leq N(C_2^{j-1})$, and i are the subscripts of boundary vertices of C_1^{j-1} , $1 \leq i \leq N(C_1^{j-1})$.

Equations (8) and (9) have been chosen with a local linear expansion classically used in marching methods. We recursively compute (8) and (9) based on vertices of curves C_1^{j-1} and C_2^{j-1} but not C_1^0 and C_2^0 . In addition, since C_1^{j-1} and C_2^{j-1} are then refined or coarsened by wavelet transforms, their resolutions are increased or reduced respectively.

5.2.2 Phase 2: Project created boundary vertices onto surface CM using a tangent plane based local approximation.

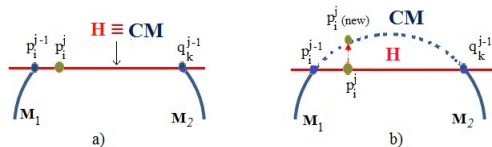


Figure 5: The connecting mesh CM created with and without using a local tangent plane approximation.

The goal of phase 2 is to improve the resulting surface CM after applying phase 1. Since new boundary vertices p_i^j and q_k^j of curves C_1^j and C_2^j are created by a

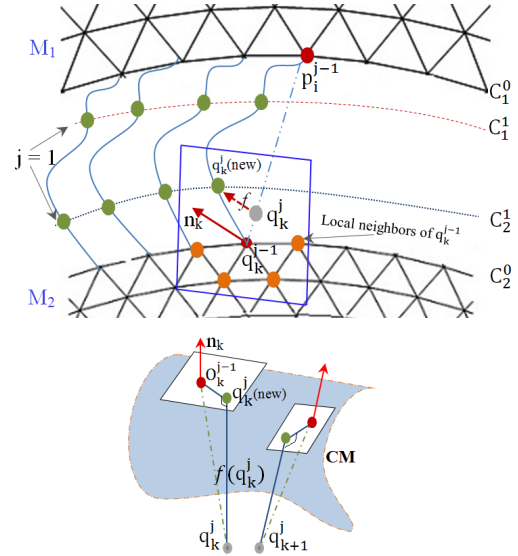


Figure 6: The vertices projected onto the surface CM.

linear interpolation in phase 1, they can lie on a flat surface H producing a flat surface CM as shown in Fig. 5a.

When CM is a complex curved surface, these newly created boundary vertices may not be on the expected surface CM because we did not consider the curvature information in phase 1. As a result, the boundary curves are produced without respect of local curvatures. Therefore, the generated connecting mesh will not respect the expected continuity between the meshes. To solve this problem, we construct the connecting surface CM by a tangent plane local approximation. We first apply phase 1 (linear interpolation) to create new boundary vertices. We then project these vertices onto tangent planes as shown in Fig. 6. Projecting the created vertices $q_k^j \in C_2^j$ onto surface CM is performed as follows: First, for each vertex q_k^j , we find the closest vertex $q_k^{j-1} \in C_2^{j-1}$ and its local neighbors $Neighbors(q_k^{j-1})$ which have edges connected to q_k^{j-1} to determine the local control vertices of q_k^j (see Fig. 7). Next, we estimate the local tangent plane $Tp(q_k^{j-1})$ of surface CM. The plane $Tp(q_k^{j-1})$ passes through the centroid vertex o_k^{j-1} (using (1)) with the unit normal vector n_k^{j-1} (using (2)). We construct the local signed

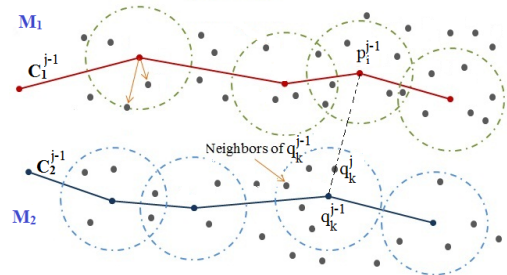


Figure 7: Selection of the local neighbors to construct a local tangent plane and a signed distance function.

distance function $f(q_k^j)$ using (3) whose value is referred to as the signed projection distance between q_k^j and $Tp(q_k^{j-1})$. Then, we use (4) to project them onto surface CM with the projection distances $f(q_k^j)$ along surface normals (see in Fig. 6). Finally, we update vertices q_k^j by their projections. We perform the same operation for vertices $p_i^j \in C_1^j$.

When the two boundary curves C_1^{j-1} and C_2^{j-1} are close together, we take the neighboring vertices from both curves to define the set of local neighboring vertices (or control vertices). For each vertex q_k^j , we keep the two closest vertices $p_i^{j-1} \in C_1^{j-1}$ and $q_k^{j-1} \in C_2^{j-1}$ with their neighbors. It permits us to take into account the local curvatures on both sides.

5.2.3 Phase 3: Refine or coarsen the new boundary curves with wavelet transforms.

Since the densities of vertices of C_1^j and C_2^j are now $N(C_1^{j-1})$ and $N(C_2^{j-1})$, we need to increase and reduce their densities to be $N(C_1^j)$ and $N(C_2^j)$. Taking advantage of the Lifted B-spline wavelet transform presented in section 3.1, we apply this transform for the multiresolution analysis of the boundary curves C_1^j and C_2^j to refine the curve C_1^j , coarsen the curve C_2^j . Then, we perform the vertex insertion or deletion operations to control the densities of vertices of C_1^j and C_2^j . Thus, the created boundary curves C_1^j , C_2^j , and the associated connecting mesh CM are changed gradually in resolution between both mesh areas.

5.3 Boundary curve connection

After creating two boundary curves C_1^j and C_2^j , we connect each new boundary curve to each previously created boundary curve, C_1^{j-1} to C_1^j and C_2^{j-1} to C_2^j , based on the method of stitching the matching borders proposed by G. Barequet et al. [Bar95]. The basic idea is the implementation of the boundary triangulation based on the distance between boundary vertices. We consider the distance between three adjacent vertices of two boundaries before connecting them together to create a triangular face (see Fig. 8). This process terminates when we reach the last vertices of both boundaries.

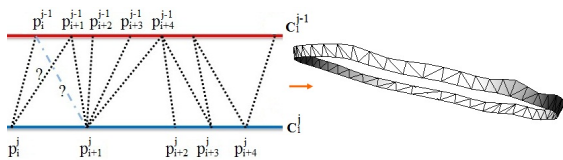


Figure 8: Figure shows a boundary curve connection.

6 RESULTS AND COMPARISONS

In this section we give some examples with experimental results to illustrate our algorithm. We also compare CM2D-TPW method with CM2D-RBFW method

which is a mesh connection method based on a RBF local interpolation and a wavelet transform. CM2D-RBFW method is built on the work by Anh-Cang Phan et al. [Phan12] in which the connecting mesh is constructed by adding triangle strips to each boundary up to the time they are close enough to be linked. This method needs to solve a linear system that additionally requires off-surface constraints to specify implicit function values. It creates off-surface points by projecting on-surface points along the surface normals with a signed projection distance d . These points are used to construct RBF support and are mandatory to obtain valid solutions. Both methods have been implemented in Matlab to make possible their comparisons. All results were obtained on a PC 2.27GHz CPU Corei5 with 3GB Ram.

In Fig. 9, CM2D-TPW algorithm produces a connecting mesh of the Tiger model which consists of two meshes defined by subdivision surfaces (Loop and Butterfly), each mesh being at a different level of subdivision. From two original coarse meshes of this model, we first apply a Loop subdivision at level 2 and a Butterfly subdivision at level 1 to obtain two meshes M_1 and M_2 of different resolutions. We then implement our algorithm to connect them together. To understand the quality of the result, we plot the image of the connecting mesh and its zoom. Based on a set of tests, $s = 4$ is an empirical good value to apply CM2D-TPW algorithm for two subdivided meshes of the Tiger model as shown in Fig. 9. From the resulting mesh, we can see that our new method can generate a smooth connecting mesh with the progressive change in resolution between meshes because it is possible to constrain the surface to have specified tangent planes at subsets of control vertices to be interpolated.

To draw comparisons, we have chosen examples of a sphere to have accurate evaluations of the error and runtime. We have developed a test on four density-based discretizations of the sphere, since analytical description permits to compute the exact surface and relative errors. The numbers of vertices are 240, 3840, 61440, 983040 and the numbers of vertices of the removed strips are 66, 720, 5982, 70743, respectively. In this way, both meshes M_1 and M_2 have the same density of vertices for a given discretization level, and the process to obtain the compatible number of vertices of CM is the same for both methods. Hence, we define the errors E_{dist} and E_{max} as follows: $E_{dist} = \sqrt{\frac{\sum_{p_i \in CM} (R - dist(c, p_i))^2}{N}}$; $E_{max} = \sup(R - d_i)$, $1 \leq i \leq N$; where: $d_i = dist(c, p_i)$ is the Euclidean distance between c and vertices p_i of CM; R, c are the radius and center of the sphere, respectively (in our tests, $c = (0, 0, 0)$ and $R = 10$). N is the number of vertices of CM.

Figs. 10-11 and Table 1 summarize the results. First, we apply CM2D-TPW algorithm for the discretiza-

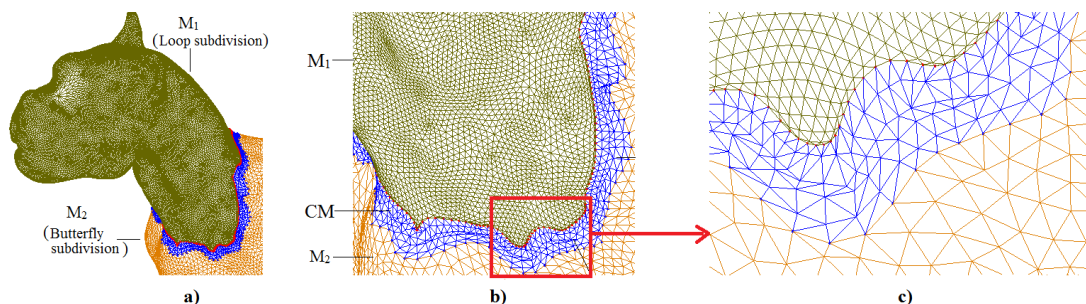


Figure 9: The Tiger model with CM2D-TPW algorithm: a) The connecting mesh CM produced with $s = 4$; b) Zoom of CM; c) Zoom of one of the interesting parts of CM.

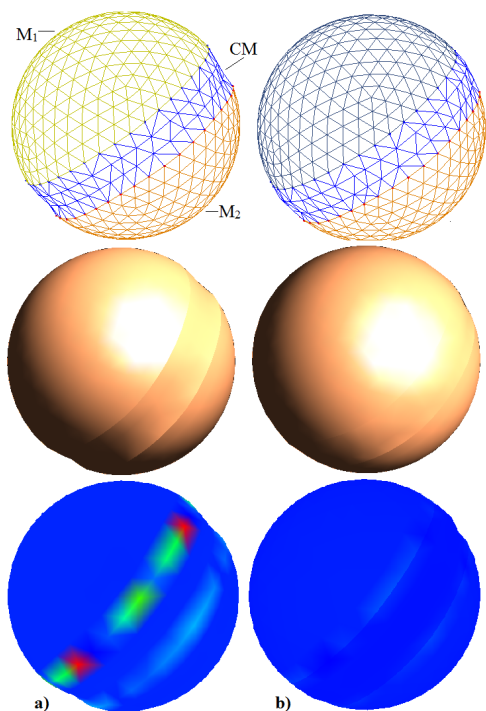


Figure 10: Model of Sphere 2: a) CM is produced by CM2D-RBFW method with $s = 2$ and $d = 0.004$; b) CM is produced by CM2D-TPW method with $s = 2$.

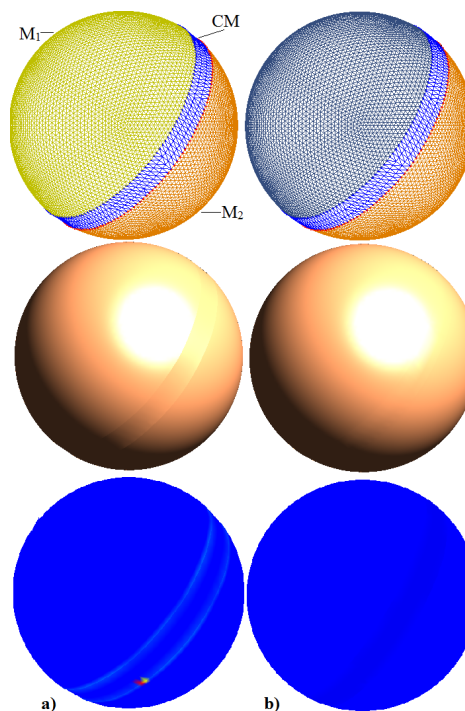


Figure 11: Model of Sphere 3: a) CM is produced by CM2D-RBFW method with $s = 2$ and $d = 0.004$; b) CM is produced by CM2D-TPW method with $s = 2$.

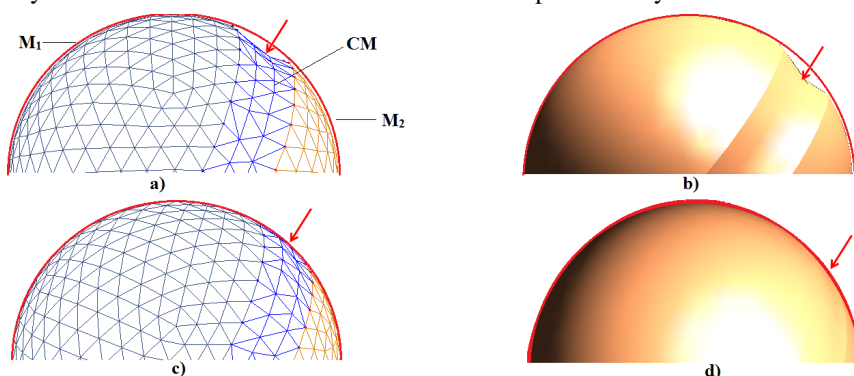


Figure 12: The surface continuity of Sphere preserved after applying CM2D-TPW method: a)-b) CM produced by linear interpolation; c)-d) CM produced by CM2D-TPW method.

tion models of the sphere with $s = 2$ as illustrated in Figs. 10b and 11b. Obviously, our method keeps the continuity between the meshes of the sphere model without destroying the Gaussian curvature and altering

the original meshes because the newly inserted vertices are on the expected surface by a local approximation with tangent plane fitting (phase 2). As a result, it gives the high quality connecting meshes and smooth

Model	CM		E_{dist}		E_{max}		Runtime (secs)	
	V	F	RBFW	TPW	RBFW	TPW	RBFW	TPW
Sphere 1	38	40	0.9265	0.7581	2.3663	2.0479	0.4061	0.3159
Sphere 2	159	240	0.2022	0.0646	0.4861	0.2216	0.4592	0.3762
Sphere 3	639	960	0.034	0.0153	0.0578	0.0293	1.3859	0.917
Sphere 4	2641	3963	0.0634	0.0329	0.1453	0.0764	14.4955	9.0221

Table 1: Comparison of errors and runtimes of CM2D-RBFW and CM2D-TPW algorithms for spheres with center $c = (0, 0, 0)$, and radius $R = 10$; the numbers of vertices and faces of CM are in columns V and F.

surfaces. Then, we also use CM2D-RBFW method on these models (see Fig. 10a and Fig. 11a).

Figs. 12a-b show the connecting mesh and surface CM produced with linear interpolation by applying phase 1 and 3 of CM2D-TPW algorithm without phase 2. As a result, CM is hyperbolic and the surface continuity is not guaranteed. While Figs. 12c-d present CM after applying all phases of the algorithm. Obviously, CM2D-TPW method generates a smooth surface with natural shape where continuity between meshes is preserved.

According to these experimental results, we can see that CM2D-TPW method gives better results compared to CM2D-RBFW method since errors to the real surface are smaller and Gaussian curvatures are much better respected. In addition, a well-known drawback of RBF based reconstruction methods is the difficulty to provide abrupt changes in a small distance (see [Luo08]). It requires much more estimation which includes estimating the linear constraints on the control vertices as well as the off-surface constraints to construct and solve a linear system for each interpolated vertex. Therefore, the time of computation will be inevitably longer or the memory requirements may exceed the capacity of the computer. As a consequence, the runtime of this algorithm is rapidly increasing when the vertex numbers of the models increase as illustrated in Table 1. We have applied the algorithm to various 3D objects with complex shapes. The runtime increases quadratically. Moreover, the most critical disadvantage is that it is very important for the user to make a decision on the choice of the basis functions and the user parameter values, i.e d -the signed distance and h -the shape parameter. This leads to the fact that the user chooses them by a rather costly trial and performs their numerical experiments over and over again until they end up with a satisfactory result consisting of the well-chosen values and an interpolated surface with a natural shape. In order to overcome these disadvantages, we have proposed a more reliable method to join two meshes. It produces surfaces of good approximation, computationally more efficient and occupied less memory compared to the C2MD-RBFW method. The memory storage will not become a problem when the numbers of vertices of the given meshes are large in practical applications. The computing time of this algorithm is smaller than CM2D-RBFW algorithm as shown in Table 1 while we

have not taken into account the execution time of experiments for values d, h in CM2D-RBFW method.

7 CONCLUSION

We have introduced a new simple and efficient mesh connection method which produces a high quality connecting mesh and finally a smooth surface. The mesh is changed gradually in resolution from one area to the other one. CM2D-TPW method joins two meshes with different resolutions while maintaining the surface continuity and not destroying local curvatures. It keeps the original boundaries of the meshes and the closest faces around these boundaries while connecting them. The advantages of this method are: 1) It is simple, efficient, and local; 2) It generates smooth connecting surfaces; 3) There is no need to solve a system of linear equations. As a consequence, our algorithm is then numerically stable. These features make CM2D-TPW method become feasible and suitable for designing, joining and modeling 3D objects with complex shapes. Thus, it can be extended to applications related to pasting meshes, and filling holes.

ACKNOWLEDGMENTS

We would like to thank all reviewers for their valuable comments which help us to improve the paper.

8 REFERENCES

- [Ale03] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE Trans. on Visualization and Comp. Graph.*,9(1):3-15,2003.
- [Ale04] M. Alexa, S. Rusinkiewicz, M. Alexa, and A. Adamson. On normals and projection operators for surfaces defined by point sets. In *Eurograph. Symp. on Point-Based Graph.*, p. 149-155, 2004.
- [Ber02] M. Bertram. Biorthogonal wavelets for subdivision volumes. In *Proc. of the SMA'02*, p. 72-82, New York, USA, 2002. ACM.
- [Ber04a] M. Bertram, M. A. Duchaineau, B. Hamann, and K. I. Joy. Generalized B-spline subdivision-surface wavelets for geometry compression. *IEEE*, 10:326-338, 2004.
- [Bra06] John Branch, Flavio Prieto, and Pierre Boulanger. Automatic hole-filling of triangular

- meshes using local Radial Basis Function. In 3DPVT, pages 727-734, 2006.
- [Bar95] Gill Barequet and Micha Sharir. Filling gaps in the boundary of a polyhedron. *Comp. Aided Geometric Design*, 12(2):207-229, 1995.
- [Car01] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *Proc. of the SIGGRAPH'01*, p.67-76, ACM, NY, USA, 2001.
- [Cat98] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes, p.183-188. ACM, NY, USA, 1998.
- [Cas05] G. Casciola, D. Lazzaro, L. B. Monte-fusco, and S. Morigi. Fast surface reconstruction and hole filling using radial basis functions, numerical algorithms, 2005.
- [Dyn90] N. Dyn, D. Levin, and J. A. Gregory. A Butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9:160-169, 1990.
- [Doo78] D. Doo and M. Sabin. Behaviour of recursive subdivision surfaces near extraordinary points. *CAD*, 10(6):356-360, 1978.
- [Dor97] Chitra Dorai, John Weng, and Anil K. Jain. Optimal registration of object views using range data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(10):1131-1138, 1997.
- [Fu04] Hongbo Fu, Chiew-Lan Tai, and Hongxin Zhang. Topology free cut and paste editing over meshes. In *GMP*, pages 173-184, 2004.
- [Hus10] N. A. Husain, A. Bade, R. Kumoi, and M. S. M. Rahim. Iterative selection criteria to improve simple adaptive subdivision surfaces method in handling cracks for triangular meshes. In *Pro. of the VRCAI'10*, p. 207-210, USA, 2010. ACM.
- [Hop92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph.*, 26:71-78, 1992.
- [Jia07] D. Jiang and N. F. Stewart. Reliable joining of surfaces for combined mesh-surface models. In *Pro. of 21st ECMS*, pages 297-303, 2007.
- [Kho00] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive geometry compression. In *Proc. of the Comp. Graph. Conf., SIGGRAPH*, pages 271-278, New York, 2000. ACM.
- [Lev03] D. Levin. Mesh-independent surface interpolation. In *Geometric Modeling for Scientific Visualization*, volume 2, pages 37-49, 2003.
- [Loo87] Charles Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, 1987.
- [Lou97] M. Lounsbery, T. DeRose, and J. D. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM*, p.34-73, 1997.
- [Luo08] W. Luo, M. C. Taylor, and S. R. Parker. A comparison of spatial interpolation methods to estimate continuous wind speed surfaces using irregularly distributed data from England and Wales. *Inter. Journal of Climatology*, 28(7):947-959, 2008.
- [Mal98] S. G. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1998.
- [Hus11] N. A. Husain, M. S. M. Rahim, and A. Bade. Iterative process to improve simple adaptive subdivision surfaces method with Butterfly scheme. In *World Academy of Science, Engineering and Tech.*, pages 622-626, 2011.
- [Ols08] L. J. Olsen and F. F. Samavati. A discrete approach to multiresolution curves and surfaces. In *Proc. of the 2008 International Conf. on Computational Sciences and Its App.*, p. 468-477, Washington, DC, USA, 2008. IEEE.
- [Phan12] Anh-Cang Phan, R. Raffin, and M. Daniel. Mesh connection with RBF local interpolation and wavelet transform. In *Pro. of the SoICT '12*, pages 81-90, New York, NY, USA, 2012. ACM.
- [Pak07] H. Pakdel and F. F. Samavati. Incremental subdivision for triangle meshes. In *Journal of Computational Science Engineering*, vol 3, No. 1, pages 80-92, 2007.
- [Sam04] F. F. Samavati. Local filters of B-spline wavelets. In *Proc. of International Workshop on Biometric Tech. 2004*, p.105-110, 2004.
- [Sar11] Scott A. Sarra. Radial basis function approximation methods with extended precision floating point arithmetic. *Engineering Analysis with Boundary Elements*, 35(1):68-76, 2011.
- [Sto96] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann Pub., 1996.
- [Suc09] N. Suciati and K. Harada. Wavelets-based multiresolution surface as framework for editing the global and local shapes. *Inter. Journal of Comp. Science and Net. Security*, 5:77-83, 2009.
- [Swe96b] W. Sweldens and P. Schröder. Building your own wavelets at home. In *Waveletes in Computer Graphics*, p. 15-87, 1996.
- [Swe98] Wim Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29:511-546, 1998.
- [Zor00] Denis Zorin and P. Schröder. Subdivision for Modeling and Animation. Technical report, SIGGRAPH 2000, Course Notes, 2000.

High-Definition Texture Reconstruction for 3D Image-based Modeling

Hoang Minh Nguyen
The University of Auckland,
New Zealand
hngu039@aucklanduni.ac.nz

Christof Lutteroth
The University of Auckland,
New Zealand
lutteroth@cs.auckland.ac.nz

Burkhard Wünsche
The University of Auckland,
New Zealand
burkhard@cs.auckland.ac.nz

Wannes van der Mark
The University of Auckland,
New Zealand
w.vandermark@auckland.ac.nz

Patrice Delmas
The University of Auckland,
New Zealand
p.delmas@cs.auckland.ac.nz

Eugene Zhang
Oregon State University,
Oregon, United States
zhange@eecs.oregonstate.edu

ABSTRACT

Image-based modeling is becoming increasingly popular as a means to create realistic 3D digital models of real-world objects. Applications range from games and e-commerce to virtual worlds and 3D printing. Most research in computer vision has concentrated on the precise reconstruction of geometry. However, in order to improve realism and enable use in professional production pipelines digital models need a high-resolution texture map. In this paper we present a novel system for creating detailed texture maps from a set of input images and estimated 3D geometry. The solution uses a mesh segmentation and charting approach in order to create a low-distortion mesh parameterization suitable for objects of arbitrary genus. Texture maps for each mesh segment are created by back-projecting the best-fitting input images onto each surface segment, and smoothly fusing them together using graph-cut techniques. We investigate the effect of different input parameters, and present results obtained for reconstructing a variety of different 3D objects from input images acquired using an unconstrained and uncalibrated camera.

Keywords

Texture reconstruction, Image-based modeling, mesh parameterization, texture mapping

1 INTRODUCTION

Digital 3D models are used in a large number of applications ranging from entertainment (games, movies) to engineering and architecture (design), e-commerce (advertisement) and education (simulation and training). 3D model creation can be made more effective, more affordable, and more accessible to inexperienced users, by using image-based reconstruction methods, which aim to create a high-quality digital model from a set of input photographs [HVC08, REH06].

Most published research has concentrated on the problem of reconstructing 3D geometry from a set of input images, and estimating camera parameters for methods assuming uncalibrated and unconstrained image acquisition. The problem of texture reconstruction for multi-view stereo has also been investigated, however, many

authors make assumptions, such as known camera parameters, which can not be guaranteed in practice.

In this paper we present a complete system for texture reconstruction for image-based modeling. The system is fully automatic and input images can be acquired with an unconstrained and uncalibrated camera. The resulting models contain a high-definition texture map and can be integrated into professional production pipelines. Our algorithm automatically estimates the intrinsic and extrinsic parameters of the input cameras using *Structure-from-Motion* and *Bundle Adjustment* techniques. The 3D model is then automatically parameterized using a segmentation and charting technique, which is suitable for surfaces of arbitrary genus [ZMT05]. A texture map is then created by back-projecting the best fitting input images onto each surface segment, and smoothly fusing them together over the corresponding chart by using graph-cut techniques.

The remainder of this paper is organized as follows. Section 2 reviews existing approaches for texture reconstruction in multi-view stereo. Section 3 summarizes our image-based modeling technology, which we use to create 3D geometry and estimate camera parameters. Section 4 describes our texture reconstruction process in detail. Section 5 evaluates our solution and discusses

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

the effect of various parameters and the algorithm's advantages and shortcomings. We conclude this paper and give an outlook on future research in section 6.

2 LITERATURE REVIEW

Image-based texture reconstruction for 3D models requires in general two steps: a surface parameterization of the reconstructed 3D object, and computation of the object's surface texture from a set of input images of the object.

The surface parameterization creates a mapping of a 2D domain (parameter space) to the surface mesh of the reconstructed 3D object. Texture mapping can then be accomplished by creating a 2D texture image over the parameter space. An explicit surface parameterization can be avoided by determining the input image regions best representing the object's surface, blending them together, and storing them in a texture atlas indexed by the mesh vertices [XLL⁺10]. However, since there is no global parameterization, postprocessing algorithms, such as polygon reduction, can result in unwanted artifacts.

Surface parameterization methods can be classified according to their complexity, whether the resulting mapping is bijective, whether they have a predetermined boundary for the parameter space, and to what extent distortion is minimized [SPR06]. For objects with a non-zero genus or complex geometry the surface must be cut into multiple parts and parameterized individually in order to minimize distortions. The resulting charts can be combined into one single texture atlas using a packaging algorithm.

Most recent image-based texture reconstruction algorithm seem to use a charting approach. Goldluecke and Cremers [GC09] create a planar texture space via an automatically created conformal atlas [LWC06]. The planar texture space is then used to solve a partial differential equation, originally defined over the object's surface, in order to find the surface texture representing the input images best.

Computation of a surface texture from input images is difficult since several images mapping to the same surface region can result in conflicting color information due to geometric errors (camera parameters), limited image resolution, and varying environmental parameters (lighting) during image acquisition. Four classes of solutions are described in the literature:

1. Blend input image information per texel using suitable weights for different source images [BMR01, LH01].
2. Compute texture patches and fuse them seamlessly together by optimizing seam locations [LI07, XLL⁺10] or warping texture patches [EdDM⁺08].

3. Compute texture patches and blend them seamlessly together. Chen et al. use multi-band blending in order to minimize seam discontinuities [CZCW12].
4. Use a local optimization step in order to fully utilize the information given by multiple images of the same object region. Goldluecke and Cremers present a technique for computing high-resolution texture maps from lower-resolution photographs [GC09]. The method requires accurate geometry and camera calibration.

Additional optimization steps are possible to take into account texture differences in input images, e.g., due to illumination changes, shadows, and camera parameters such as dynamic range adjustment. Xu et al. [XLL⁺10] use radiometric correction to adjust color difference between patches. Valkenburg and Alwesh reduce seams resulting from image illumination variations by applying a global optimization to all vertex colors of a 3D mesh [VA12]. Chen et al. remove highlight effects by determining all input images mapping to a surface area [CZCW12]. Image regions which vary too much from the median color of the surface area are removed. Missing or deleted image regions (e.g., highlights) can be filled using Poisson image editing [CZCW12, CAH⁺13].

3 3D GEOMETRY RECONSTRUCTION

In this section we summarize our image-based modeling algorithm for geometry reconstruction. We concentrate on the algorithm steps effecting texture reconstruction, i.e., camera parameter estimation and surface representation. More details of the algorithm are described in [NWDL13, NWDL12b].

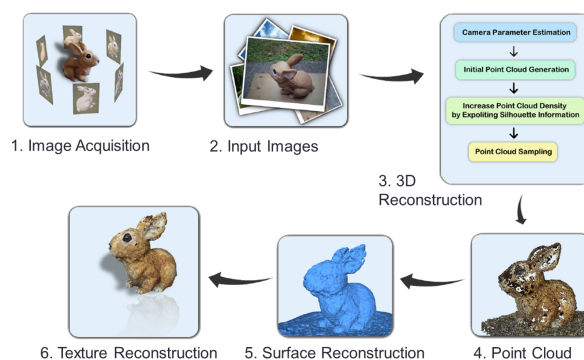


Figure 1: Overview of our algorithm for reconstructing 3D models from a set of unconstrained and uncalibrated images.

An overview of our image-based modeling technology is given in Figure 1. The algorithm uses a coarse-to-fine strategy where a rough model is first reconstructed and then sequentially refined through a series of steps.

The first step of the geometry reconstruction consists of estimating the camera parameters for each view. This is accomplished by detecting and extracting distinctive features using a SIFT feature detector [Low99, Low04]. We then isolate all matching images, selecting those that contain a common subset of 3D points [HQZH08]. Given a set of matching images, a scene geometry (point cloud) and camera pose can be estimated simultaneously by using a *Structure from Motion* algorithm and subsequently refining the solution using *Bundle Adjustment*. The last step is critical for the accuracy of the reconstruction, as concentration of pairwise homographies would accumulate errors and disregard constraints between images. The method minimizes the reprojection error, which is defined by the distance between the projections of each point and its observations.

Due to the sparseness of the point cloud representing the scene geometry, artifacts can arise during the surface and texture reconstruction processes. We overcome this problem by integrating a shape-from-silhouette approach. Silhouette data is obtained by using the rough depth estimation from the previous step for a foreground segmentation and applying the Marching Squares algorithm [Lor95]. The complexity of each silhouette line is reduced using the Douglas-Peucker algorithm [VW90]. The 3D positions of silhouette points are estimated by forming cone lines from silhouette contour points and the camera's estimated optical center, projecting the lines onto the other silhouettes, computing the intersection points, and lifting them to 3D [MBR⁺00].

Adding silhouette points and using them in the bundle adjustment step results in a better camera parameter estimation and smoother surface reconstruction.

Finally the object's surface is reconstructed. We tested the α -shape algorithm, the power crust algorithm, and the ball pivot algorithm. In the end we decided to use the Poisson surface reconstruction algorithm [KBH06]. The technique gives a smoother reconstruction than other tested techniques, is more stable towards noise, and always creates a watertight surface.

A perceived weakness of the algorithm is that it requires oriented normals at the input points. However, we can obtain them from the image and silhouette information. Furthermore, it has been shown that the approach is quite resilient to inaccuracies in the directions of the normals [Kaz05].

A surface texture is created by projecting each vertex of the mesh onto all input images containing the point (i.e., the surface point is visible from the images' estimate camera location). The mesh vertex color is the weighted average of the corresponding image pixels. The resulting triangle mesh with vertex colors is rendered using Gouraud shading. An example is shown in Figure 2.

Color interpolation suffers from two major shortcomings: (1) detailed input image textures appear blurred (see bottom row of Figure 2), and (2) texture resolution is lost if a mesh reduction method is applied.



Figure 2: Photograph of a rooster statue (left) and the reconstructed model using vertex colors and Gouraud shading (right). The images at the bottom show an enlargement of the neck region of the object.

4 TEXTURE RECONSTRUCTION

We create a high quality texture map for our 3D model in two steps: The 3D mesh model is first parameterized yielding a one-to-one triangle mapping from the 3D model to a 2D planar surface. Input images are then projected onto the surface and suitable texture regions are identified, cut, and fused together to form a 2D texture atlas.

4.1 Surface Parameterization

The objective is to segment the resulting meshes into patches and unwrap them onto a 2D planar surface. We evaluated different surface parameterization techniques, but found that existing libraries, such as Blender, either create a very disjoint map of triangle patches, or create a single parameter patch with large distortions. We hence use a *Feature-based Surface Parameterization*, which consists of three stages [ZMT05]: Genus reduction, feature identification, and patch creation.

Genus reduction In order to identify non-zero genus surfaces, a surface-based *Reeb* graph [Ree46] induced

by the *average geodesic distance* [HSKK01] is constructed. The leaf nodes of this graph reveal the tips of the protrusions of the meshes, while loops in the graph signify the existence of handles. The principle behind genus reduction is to identify loops that do not separate the surface into two disjoint connected components and cut the surface open along the cycle, which reduces the combined genus of the surface segments by one. This process is repeated until there are no more handles.

Feature identification From the *Reeb* graph the tips of protrusions are identified and the features are separated from the rest of the surface by constructing a closed curve γ as follows: We separate the region R that corresponds to the tip of the protrusion by first computing the function $f_p(\mathbf{q}) = g(\mathbf{p}, \mathbf{q})$, where $f_p(\mathbf{q})$ is the *geodesic distance function* [HSKK01] with respect to \mathbf{p} . The value of f_p is normalized to fit in the interval $[0, 1]$. Regions which are bounded by a given *isovalue* are examined. Specifically, the interval $[0, 1]$ is partitioned into k equal sections. The surface is then divided into levelset bands by performing region-growing from the tip of the protrusion \mathbf{p} based on the values of f_p in these intervals [ZMT05].

Variation in the *area* of this sequence of bands tends to be small along a protrusion slope, and large where the feature connects to the remaining section of the surface. The separating region R can be extracted by examining these areas, which are considered as a continuous function $\mathbf{A}(x)$. To remove any small undulations, $\mathbf{A}(x)$ is passed through a *Gaussian filter* function N times.

Three parameters (*isovalue*, k , and N) influence the effectiveness and efficiency of the region separation process. The larger the *isovalue* is, the further the region-growing process continues. This leads to fewer surface patches being generated. Higher k values result in more samples being used to discretize $\mathbf{A}(x)$, increasing the probability of small noise being considered as potential candidate places for the separating region. Large N values tend to cause the location of the separating region to shift or it being lost, while too small values often result in false separations.

Once the separating region R has been identified, a closed curve γ separating the surface into segments is constructed as follows: A collection of edges in the surface separating the feature from the rest of the surface (the skeleton) of R is found. During this process dangling edges are rejected. A separating cycle ρ from this skeleton is then extracted. Finally, a shorter and smoother separating cycle γ is constructed based on ρ .

Patch creation Patches are created by unwrapping them using a *discrete conformal mapping* [EDD⁺95]. The method creates first texture coordinates of the boundary vertices, and then determines texture coordinates of the interior vertices through solving a closed form system. The main problem with this

mapping technique is that regions can be stretched or compressed during the process leading to areas of the meshes not being preserved. This in turn results in uneven sampling rates across the surface.

Interior vertices' texture coordinates are optimized to reduce the geometric distortion by first computing an initial harmonic parameterization [Flo97]. A *square virtual boundary* enclosing the patch is constructed. The exact coordinates of the boundary are not important as long as they do not coincide with those of the patch boundary. We then perform triangulation of the regions between the virtual boundary and the original boundary using Scaffold triangles. The patch optimization technique proposed by Sandle *et al.* [SGSH02] is then applied to the enlarged patch.

4.2 Texture Map Generation

At this stage, we have successfully generated a parameterization of the 3D model. The next task is to construct a complete texture map using the computed parameterization. This is accomplished in three steps:

1. Identify images and regions of input images to be mapped onto each patch of the parameterization.
2. Cut these patches and paste them over the parameterized surface.
3. Merge overlapping regions using a *graph cut* technique [KSE⁺03a, CFW⁺12].

Texture region identification: For each patch of the surface parameterization we need to identify the image regions mapping onto it. We project all triangles of a patch onto all input images where it is visible, i.e.: (1) the triangle normal forms an angle of less than 90° with the vector to the estimated camera position; (2) the triangle is not occluded by other surface regions. The resulting image regions and the one-to-one correspondence between projected triangles and original triangles of the patch is saved for the next stage of the algorithm.

Texture map computation: At this stage for each patch we have a set of texture regions. The goal is to process these texture regions to produce a new texture that will cover the patch. We perform the mapping of a texture region from an input image to a patch for each triangle separately. Given two arbitrary triangles Δ_1 and Δ_2 , an affine transformation that transforms triangle $\Delta_1(P, Q, R)$ to $\Delta_2(P^\circ, Q^\circ, R^\circ)$ is defined as follows: Let Φ_1 be the affine transformation that maps the unit triangle to Δ_1 , and Φ_2 be the affine transformation that maps the unit triangle to Δ_2 . The affine equivalence of these two triangles is $\Phi_2 \circ \Phi_1^{-1}$.

The procedure is repeated for each texture region yielding a set of overlapping textures covering the face of the

processed patch. We use a greedy technique to assemble these textures. We start with the least fitting texture and project it onto the input image. We then use the next least fitting texture and add as much as possible of it while minimizing the seam between the two textures using a *graphcut technique* [KSE⁺03a]. This process is repeated until all input images have been considered. The effect of this strategy is that artifacts which occur only in one input image, such as highlights, are reduced since frequently they result in a visible seam with the current partial texture map. Furthermore the last texture added is the one from the best fitting input image, so most of the final texture results from this image unless it creates inconsistencies with the other input images. Note that the current method does not guarantee removal of artifacts. For example, if a surface region is only visible in one input image and it contains a highlight, then this highlight is part of the final texture map. We have tested this algorithm with more than 40 data sets and did not encounter any problems apart from the shading inconsistencies explained in subsection 5.2.3.

Seam Minimization: Seams between overlapping input image texture regions are minimized by using a *graphcut technique* [KSE⁺03a]. Given two overlapping images A and B , we want to find the cut within the overlap region, which creates the best transition between these images. The overlap region is represented as directed graph, where each node represents a pixel position p in the overlap region, which is denoted $A(p)$ and $B(p)$ for the two images A and B , respectively. Nodes are connected by edges representing 4-connectivity between pixels. Each edge is given a cost encoding the pixel differences between the two source images at that position.

We have investigated the effect of different parameters for image fusion applications [CFW⁺12] and tested them with various 3D models. Based on this we use the following parameters: Image pixels are represented in the RGB color space. Color distances are computed using the L_2 norm. The cost function w corresponds to the gradient weighted color difference between the images A and B at the neighboring pixels p and q , i.e.,

$$w_{\nabla} = \frac{\|A(p) - B(p)\| + \|A(q) - B(q)\|}{\|G_A^{pq}(p)\| + \|G_A^{pq}(q)\| + \|G_B^{pq}(p)\| + \|G_B^{pq}(q)\|}$$

where $G_A^{pq}(p)$ is the image gradient in the direction of the edge pq at pixel p . This cost function has been originally devised by Kwatra et al. [KSE⁺03b] based on the observation that seams are more noticeable in low-frequency regions, and a visually more pleasing cut is computed by increasing the cost of an edge with a decreasing image gradient.

Figure 3 illustrates an example in which two texture patches of our *Rooster* model are fused together to form a larger and more complete texture patch. The newly

merged texture patch is then fused together with the next available texture patch in the list. The process terminates when all texture patches have been successfully merged.

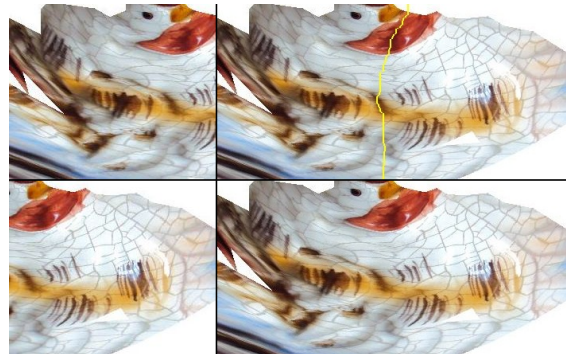


Figure 3: Seam minimization. Source texture patches are shown in the left column, while the merged texture patch is shown in the right column.

Figure 4 shows the texture map obtained by back-projection surface patches onto the input images (right) and the resulting textured 3D model (left). In many instances the input images do not cover the entire surface of the object. For example, in many of our experiments users did not take photos of the underside of objects. In this case the 3D point cloud contains large gaps. The Poisson surface reconstruction will create a smooth watertight surface interpolating the gaps, but the corresponding regions of the texture map have no color information (red color regions in the top-right image of Figure 4). The accuracy of our new texture reconstruction process is illustrated by comparing the bottom-left image of Figure 2 and the bottom-right image of Figure 4.

5 RESULTS

5.1 Effect of Parameters

We have investigated the effect of different algorithm parameters on the quality of the surface parameterization and texture reconstruction.

5.1.1 Isovalue

The larger the *isovalue* is, the farther the region-growing process continues, and the fewer surface patches are generated. Figure 5 illustrates the surface segmentation and Figure 6 the resulting texture patches. If the isovalue is too large the resulting texture map suffers from large distortions. However, having a single texture patch simplifies some operations such as image inpainting to fill surface regions without matching input images.

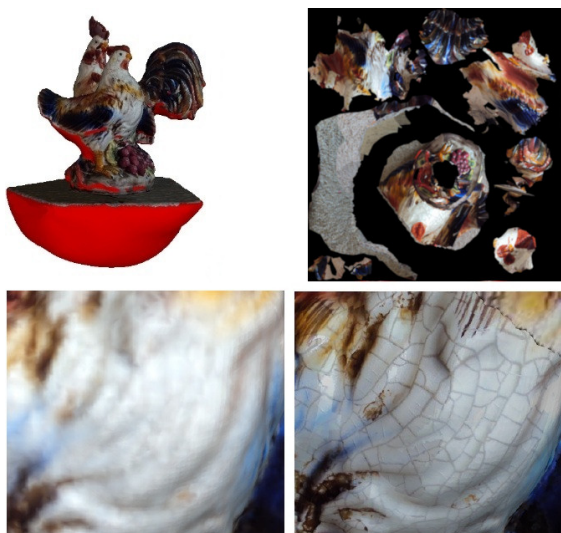


Figure 4: Top row: Reconstruction of the *Rooster* in Figure 2 (left) and the surface parameterization after texture map computation (right). Regions that were not visible in any of the input images are colored red. Bottom row: Surface appearance of the rooster’s neck region using vertex color interpolation (left) and our new texture reconstruction process (right).

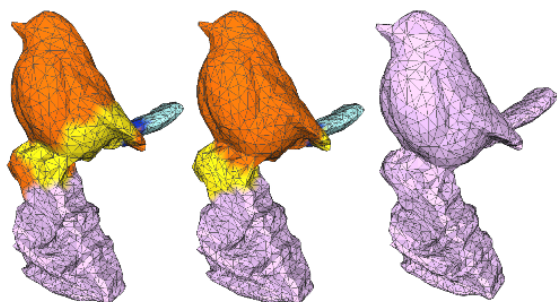


Figure 5: Parameterization of our *Bird* model with the isovalues of 1.0, 2.0, and 5.0, respectively.

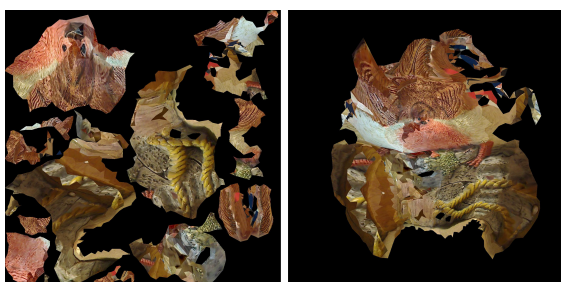


Figure 6: Texture map for the surface parameterization obtained using an isovalue of 2.0 (left) and 5.0 (right).

5.1.2 Number of Gaussian Iteration Steps

Increasing the number of times the *Gaussian filter* function is applied during the parameterization process, effects how sensitive the segmentation process is towards differently sized features. Figure 7 demonstrates that small values result in unnecessarily many segments,

whereas large values result in too few patches and hence larger texture distortions.

Figure 8 shows that the resulting texture maps look very similar. However, the texture map generated using 10 Gaussian steps contains falsely oriented texture features in the neck region of the bird model. This seems to be due to aliasing effects caused by a high distortion of the corresponding parameter space region. Contributing causes are the relatively low resolution of the web-cam images, and the fact that we currently use a nearest neighbor interpolation for the texture reconstruction.

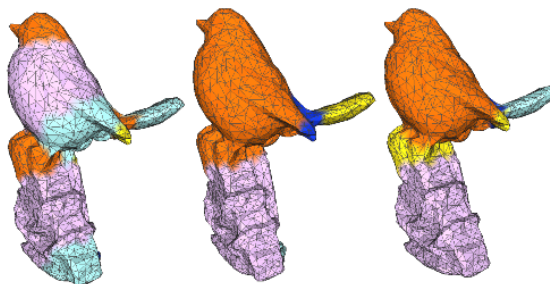


Figure 7: Parameterization of the *Bird* model with (from left to right) 10, 30, and 50 Gaussian steps, respectively.

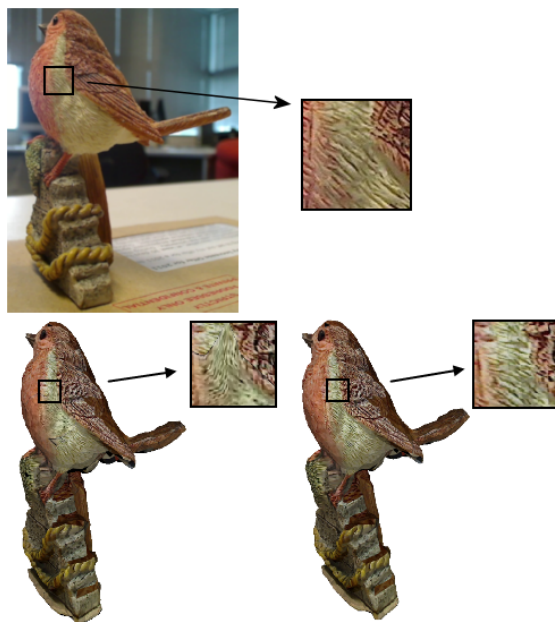


Figure 8: Top: an input image of the *bird* data set. Bottom: the texture map created using 10 (left) and 30 (right) Gaussian steps.

5.2 Reconstruction Results

We have evaluated our system using a variety of datasets of objects at different scales acquired under different weather and lighting conditions. In general, our system produces qualitatively good results with high resolution textures for both uniformly colored and feature-poor objects, and for objects with concave

regions and moderately complex geometries. The size of our test datasets varied from as few as 6 images to hundreds of images. All input images were acquired with simple consumer-level handheld cameras, including a Smartphone camera. Our systems fails for objects which have viewpoint dependent surface appearance, e.g., refractive and reflective materials within complex environments. This section contains a summary of different experiments that we performed to evaluate our texture reconstruction method.

5.2.1 Rooster Dataset

The first dataset contains 35 images of a *White Rooster* with a resolution of 2592×1944 pixels. Figure 9 shows some of the input images. The original object has a complex surface geometry with many bumps and wrinkles. Notice that most of the surface of the model contains few visual features.



Figure 9: Two out of 35 input images of the *White Rooster* datasets.

The resulting reconstructed model, shown in the left of Figure 13, is of good quality and bears a high resemblance to the original object. The overall shape, along with details such as feathers of the original model are reconstructed well. The resulting model consists of 298,187 polygons. There are a few regions (underneath the model) where no texture has been generated (colored in red) due to missing input images showing these regions.

5.2.2 General Dataset

This data set contains 18 images (2592×1944 pixels resolution) of a *General* figurine. The original model has a very smooth, reflective and shiny surface. The reconstruction, shown on the right-hand side of Figure 10, is of good quality and the final model has a high resemblance to the original object. The resulting model consist of 101,778 polygons. The texture is very realistic, but contains some visible seams along patch boundaries



Figure 10: Input image of the *General* dataset (left) and the resulting reconstruction (right).

5.2.3 Vase Dataset

This dataset contains 26 images (2592×1944 pixels resolution) of a vase. The original object has a very smooth, reflective and shiny surface with repetitive textures. The reconstructed model has 215,918 polygons. The geometry of the reconstruction is very realistic. However, the texture reconstruction shows some visible illumination differences due to some input images having been taken with flash and some without. In future we plan to overcome these problems by using multi-band blending techniques [APK08] and global optimization of luminance values in CIELUV color space along seam boundaries.



Figure 11: Image of a vase (left) and the resulting 3D reconstruction (right). The enlargement shows brightness variations due to some input images taken with flash.

5.2.4 Objects with High Genus

Section 2 reviewed previously presented techniques for texture reconstruction. Despite some seemingly impressive results, we did not find any examples in the literature for objects with high genus, for which geometry and texture reconstruction are notoriously difficult. Figure 12 illustrates that our image-based modeling system

and texture reconstruction method handles such cases without problems.



Figure 12: Two examples of models with a high genus: input image (top), 3D reconstruction (middle), and the surface parameterization (bottom).

5.3 Running Time

The presented algorithm has not been optimized yet and the running time varies between approximately 10 minutes for the reconstruction of an apple from 6 photographs, to many hours for more complex models. For example, the reconstruction of the rooster data set in subsection 5.2.1 takes 6 hours and 19 minutes on a PC with Intel Quad Core i7 CPU and 6GB RAM. The time requirements of the various stages of the algorithm are:

1. Camera Parameter Estimation: 18.6% = 71 minutes (feature detection and matching are implemented in parallel and use all four cores of the CPU)
2. Point Cloud Generation: 33.0% = 125 mins
3. Mesh Processing: 9.8% = 37 mins
4. Texture Reconstruction: 38.6% = 146 minutes

Initial tests indicate that a GPU implementation would be 50-100 times faster. Alternatively a compute cloud could be used to speed up computation.

5.4 Comparison

The combination of “Bundler” [SSS08] and CMVS & PMVS [FCSS10] is a well-known and open-source

image-based modeling system. However, the output of these research tools is a dense point cloud. While we can easily obtain a closed surface from this data, we were unable to find published software for texture reconstruction. We hence compared our system with the only complete systems we could find. We identified thirteen companies working in this field and compared the best four algorithms [NWDL12a]. We showed that our solution and “123D Catch” achieved the best geometry reconstruction. The system presented in this paper achieves even higher quality reconstructions due to the integration of silhouette information and the novel texture reconstruction algorithm. Figure 13 demonstrates that these improvements make a significant difference when dealing with data sets containing few distinct visual features. For such data sets “123D Catch” struggles both with reconstructing a correct geometry and appropriate texture map.



Figure 13: 3D reconstruction from the “white rooster data set” using our method (left) and “123D Catch” (right).

6 CONCLUSION AND FUTURE WORK

We have described a texture reconstruction technique for image-based modeling systems. In contrast to previously presented methods we integrate shape-from-silhouette and correspondence-based methods, which gives us very reliable camera parameter estimates and excellent geometry reconstruction. This enables us to fuse together texture regions obtained from input images without requiring excessive blending and deformations. Textures are combined using a greedy algorithm and a graph-cut technique minimizing gradient weighted color differences. The texture reconstruction uses an advanced surface parameterization method which takes into account the genus and geometric features of an object. We have demonstrated the quality of the reconstruction process using objects with different geometries, genus, colors and surface properties. In all cases we achieved an excellent reconstruction and realistic texture. In contrast to laser scanners our system also works for shiny and dark objects, and is easily scalable.

Some problems still exist with seams along texture patches, and discontinuities due to color inconsistencies created during the image acquisition process. The current system does not generate a texture for surface regions not visible in the input images. We currently work on texture inpainting techniques and exemplar-based texture synthesis to fill such regions [PGB03, CPT04].

7 REFERENCES

- [APK08] Cedric Allene, Jean-Philippe Pons, and Renaud Keriven. Seamless image-based texture atlases using multi-band blending. *19th International Conference on Pattern Recognition*, pages 1–4, 2008.
- [BMR01] Fausto Bernardini, Ioana M. Martin, and Holly Rushmeier. High-quality texture reconstruction from multiple scans. *IEEE Trans. on Visualization and Computer Graphics*, 7(4):318–332, October 2001.
- [CAH⁺13] A. Colburn, A. Agarwala, A. Hertzmann, B. Curless, and M.F. Cohen. Image-based remodeling. *IEEE Transactions on Visualization and Computer Graphics*, 19(1):56–66, 2013.
- [CFW⁺12] Xiao Bao Clark, Jackson Finlay, Andrew Wilson, Keith Milburn, Minh Hoang Nguyen, Christof Lutteroth, and Burkhard C. Wünsche. An investigation into graphcut parameter optimisation for image-fusion applications. In *Proceedings of Image and Vision Computing New Zealand (IVCNZ 2012)*, pages 480–485, Dunedin, New Zealand, 2012.
- [CPT04] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *Trans. Img. Proc.*, 13(9):1200–1212, September 2004.
- [CZCW12] Zhaolin Chen, Jun Zhou, Yisong Chen, and Guoping Wang. 3d texture mapping in multi-view reconstruction. In *Advances in Visual Computing*, volume 7431 of *Lecture Notes in Computer Science*, pages 359–371. Springer Berlin Heidelberg, 2012.
- [EDD⁺95] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppey, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. *Computer Graphics Proceedings (SIGGRAPH 1995)*, pages 173–182, 1995.
- [EdDM⁺08] Martin Eisemann, Bert de Decker, Marcus A. Magnor, Philippe Bekaert, Edilson de Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. Floating textures. *Computer Graphics Forum*, 27(2):409–418, 2008.
- [FCSS10] Y. Furukawa, B. Curless, S.M. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *Proceedings of Computer Vision and Pattern Recognition (CVPR 2010)*, pages 1434–1441, 2010.
- [Flo97] Michael S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, April 1997.
- [GC09] B. Goldluecke and D. Cremers. Super-resolution texture maps for multiview reconstruction. In *Proceedings of the 12th International Conference on Computer Vision (ICCV 2009)*, pages 1677–1684, 2009.
- [HQZH08] Shaoxing Hu, Jingwei Qiao, Aiwu Zhang, and Qiaozhen Huang. 3d reconstruction from image sequence taken with a hand-held camera. *International Archives of the Photogrammetry*, 37(91):559–563, 2008.
- [HSKK01] Masaki Hilaga, Yoshihisa Shinagawa, Taku Komura, and Tosiyasu L Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. *Computer Graphics Proceedings (SIGGRAPH 2001)*, pages 203–212, 2001.
- [HVC08] Carlos Hernandez, George Vogiatzis, and Roberto Cipolla. Multi-view photometric stereo. *IEEE Transaction on Pattern Recognition and Machine Intelligence*, 30(3):548–554, 2008.
- [Kaz05] Michael Kazhdan. Reconstruction of solid models from oriented point sets. In *Proc. of the 3rd Eurographics symposium on Geometry processing*, pages 73–82, 2005.
- [KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the 4th Eurographics symposium on Geometry processing*, pages 61–70, 2006.
- [KSE⁺03a] Vivek Kwata, Arno Schodl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transaction Graphics*, 22(3):277–286, 2003.
- [KSE⁺03b] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph.*,

- 22(3):277–286, July 2003.
- [LH01] Hendrik P. A. Lensch and Wolfgang Heidrich. A silhouette-based algorithm for texture registration and stitching. *Graphical Models*, 63(4):245–262, 2001.
- [LI07] V. Lempitsky and D. Ivanov. Seamless mosaicing of image-based texture maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, pages 1–6, 2007.
- [Lor95] W. E. Lorensen. Marching through the visible man. In *Proceedings of IEEE Visualization '95*, pages 368–373, 1995.
- [Low99] David G. Lowe. Object recognition from local scale-invariant features. *International Conference on Computer Vision*, 2:1150–1157, 1999.
- [Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.
- [LWC06] L. M. Lui, Y. Wang, and T. F. Chan. Solving PDEs on manifold using global conformal parameterization. In *Proceedings of the Third International Workshop on Variational, Geometric, and Level Set Methods in Computer Vision (VLSM 2005)*, pages 309–319, 2006.
- [MBR⁺00] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. In *Computer Graphics Proceedings (SIGGRAPH 2000)*, pages 369–374, 2000.
- [NWDL12a] Minh Hoang Nguyen, Burkhard C. Wünsche, Patrice Delmas, and Christof Lutteroth. 3d models from the black box: Investigating the current state of image-based modeling. In *WSCG 2012 Communication Proceedings*, pages 249–258, Pilsen, Czech Republic, June 2012.
- [NWDL12b] Minh Hoang Nguyen, Burkhard C. Wünsche, Patrice Delmas, and Christof Lutteroth. Modelling of 3d objects using unconstrained and uncalibrated images taken with a handheld camera. In *Computer Vision, Imaging and Computer Graphics - Theory and Applications*, pages 1–16. Springer Verlag, 2012.
- [NWDL13] Hoang Minh Nguyen, Burkhard Wünsche, Patrice Delmas, and Christof Lutteroth. A hybrid image-based modelling algorithm. In *Proc. of the 36th Australasian Computer Science Conference (ACSC 2013)*, pages 115–123, Adelaide, Australia, 2013.
- [PGB03] Patrick Perez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transaction Graphics*, 22(3):313–318, 2003.
- [Ree46] Georges Reeb. Sur les points singuliers d'une forme de pfaff complètement intégrable ou d'une fonction numérique [on the (singular) points of a completely integrable pfaff form or of a numerical function]. *Comptes Rendus Acad. Sciences Paris* 222, pages 847–849, 1946.
- [REH06] Fabio Remondino and Sabry El-Hakim. Image-based 3d modelling: A review. *The Photogrammetric Record*, 21:269–291, 2006.
- [SGSH02] Pedro V Sander, Steven J Gortler, John Snyder, and Hugues Hoppe. Signal-specialized parameterization. *Proceedings of the 13th Eurographics Workshop on Rendering*, pages 87–100, 2002.
- [SPR06] Alla Sheffer, Emil Praun, and Kenneth Rose. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis.*, 2(2):105–171, January 2006.
- [SSS08] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Modeling the world from internet photo collections. *Int. J. Comput. Vision*, 80(2):189–210, November 2008.
- [VA12] Robert Valkenburg and Nawar Alwesh. Seamless texture map generation from multiple images. In *Proc. of the 27th Conference on Image and Vision Computing New Zealand, IVCNZ '12*, pages 7–12, New York, NY, USA, 2012. ACM.
- [VW90] M. Visvalingam and J. D. Whyatt. The Douglas-Peucker algorithm for line simplification: re-evaluation through visualization. *Computer Graphics Forum*, 9(3):213–228, September 1990.
- [XLL⁺10] Lin Xu, E. Li, Jianguo Li, Yurong Chen, and Yimin Zhang. A general texture mapping framework for image-based 3d modeling. In *Proc. of the 17th IEEE International Conference on Image Processing (ICIP 2010)*, pages 2713–2716, 2010.
- [ZMT05] Eugene Zhang, Kobstantin Mischaikow, and Greg Turk. Feature-based surface parameterization and texture mapping. *ACM Trans. Graph.*, 24(1):1–27, 2005.

A Motion-aware Data Transfers Scheduling for Distributed Virtual Walkthrough Applications

J. Pribyl
 Dept. of Computer
 Graphics and
 Multimedia
 Faculty of
 Information
 Technology, BUT
 612 66, Brno, Czech
 Republic
 pribyl@fit.vutbr.cz

P. Zemcik
 Dept. of Computer
 Graphics and
 Multimedia
 Faculty of
 Information
 Technology, BUT
 612 66, Brno, Czech
 Rep.
 zemcik@fit.vutbr.cz

T. Burian
 Cadwork Informatik
 CI AG
 Cadwork, s.r.o.,
 Czeach team Brno
 602 00, Brno, Czech
 Rep.
 burian@cadwork.cz

B. Kudlac
 Cadwork Informatik
 CI AG
 Cadwork, s.r.o.,
 Czeach team Brno
 602 00, Brno, Czech
 Rep.
 kudlac@cadwork.cz

ABSTRACT

Data transfers scheduling is an important part of almost all distributed virtual walkthrough (DVW) applications. Its main purpose is to preserve data transfer efficiency and render quality during scene exploration. The most limiting factors here are network restrictions such as low bandwidth and high latency. Current scheduling algorithms use multi-resolution data representation, priority determination and data prefetching algorithms to minimize these restrictions. Advanced priority determination and data prefetching methods for DVW applications use mathematic description of motion to predict next position of each individual user. These methods depend on the recent motion of a user so that they can accurately predict only near locations. In the case of sudden but regular changes in user motion direction (road networks) or fast moving user, these algorithms are not sufficient to predict future position with required accuracy and at required distances. In this paper we propose a systematic solution to scheduling of data transfer for DVW applications which uses next location prediction methods to compute download priority or additionally prefetch rendered data in advance. Experiments show that compared to motion functions the proposed scheduling scheme can increase data transfer efficiency and rendered image quality during scene exploration.

Keywords

distributed virtual walkthrough, next location prediction, motion function, Markov chain, prefetching, virtual environments

1 INTRODUCTION

The initial purpose of DVW applications was to realize a virtual tourism task which allows users to visit places of interests without physically entering them (like Google Street View).

Advances in graphic and computing performance of mobile devices, sharp growth in their market and various digital media data archives created commercially or community contributed, further increase potential and usage of DVW applications. Compared to classical desktop computers, content explored within mobile

devices can be associated with richer context, like location, weather, traffic, etc.

Applications with high potential in this field are augmented reality tourist guide called LifeClipper, mobile augmented reality application Nokia City Lens, or intelligent navigations such as AIDA. Instead of focusing solely on determining routes to a specified target, the AIDA system utilizes analysis of driver behavior to identify a set of goals the driver would like to achieve (e.g. business or shopping districts, tourist areas, or real-time event information related to traffic).

AIDA visualizes all the data in a 3D scene which can help the driver understand better and interpret the delivered information. Based on the driver's motion, both the visualized information and 3D data for rendering 3D scene are downloaded on demand from a remote server via wireless connection.

The main bottleneck of DVW applications is network connection with restrictions, for example low bandwidth or higher latency, especially on wireless networks, so that transferred data can not be received by

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

clients in time. Scheduling algorithms can reduce the impact of these restrictions with the help of multi-resolution data representation, priority determination and data prefetching algorithms so that they can increase quality of rendered scene and data transfer efficiency.

Current scheduling algorithms for DVW applications widely use mathematic description of motion (motion functions) to predict next motion of each individual user. The predicted position can be used to compute download priority or to prefetch scene parts in advance. Unfortunately, these methods are accurate only for prediction of near future positions, and their accuracy decreases also in the case of sudden changes in user motion direction. For networks with higher latency, low bandwidth or just for fast moving user a prediction method with higher accuracy enabling to predict farther-positions is needed to keep data transfer efficiency and scene quality as high as possible. From another point of view, for some applications the scene quality can be a much more important parameter than the data transfer efficiency.

In this paper, we propose a systematic solution to scheduling of data transfer for DVW applications which uses next location prediction (NLP) methods which increases both data transfer efficiency and quality of rendered scene. Our solution is based on two key insights. First, NLP methods have much higher prediction accuracy compared to motion functions. Second, NLP methods can be adaptively constructed according to the multi-resolution data representation. This feature allows the scheduling algorithm to prefetch missing data at specified resolution as is required by a rendering algorithm.

2 RELATED WORK

This section briefly introduces state-of-the-art scheduling mechanisms for DVW applications.

2.1 Visibility determination

Scheduling methods for DVW applications widely use area of interest (AOI) determination algorithms [sch96], [hes98], [chi98], [li04]. Instead of downloading complete scene, it is sufficient to transfer only data in spherical area around an observer. Objects inside this area can be regarded as objects from potentially visible set (PVS) with high download priority. Wang et al.[wan09] additionally divide AOI to sections with different download priority taking into account view frustum and distance from the observer. The AOI based scheduling methods are not suitable for more complex scenes such as terrains. Marvie et al. [mar11] use PVS to schedule data transfers for complex virtual scene divided to cells by a regular grid. Download priority of PVS of adjacent cells is

determined by simple ray-casting method based on last two viewpoints. The visibility determination is also used to eliminate transferring scene parts not visible to an observer.

2.2 Motion function

Scheduling algorithms based on motion functions use vector representation of object motion, position and direction. Motion functions can be classified into linear and nonlinear [tao04], which are more accurate than the linear methods. Chim [chi98] proposes exponential weighted moving average (EWMA) motion prediction scheme which assigns different weights to past motion vectors where more recent vectors have higher weights. CyberWalk [chi03] use the EWMA scheme to achieve at least a minimum resolution of the scene. Scheduling algorithm proposed in [tel01] selects objects to be sent to client device based on integral of a benefit measure along predicted path. The prediction is made at server and is based on the assumption that once a particular type of motion is started, it will continue in the near future. This approach does not consider any previous positions. A motion-aware approach which uses state-of-the-art recursive motion function [tao04] for efficient evaluation of continuous queries on 3D object databases is described in [ali10]. The predicted positions here are used to determine download priorities of progressively recorded objects inside a virtual scene so that only exact portion of each object will be downloaded based on the computed priorities. In [sch06] an algorithm for speculative prefetching of terrain tiles is presented. It predicts viewpoint motion by fitting a spline through a list of last positions so the tiles that are predicted to become visible in the near future can be prefetched in advance.

2.3 Next location prediction

Next location prediction (NLP) methods make the assumption that there is a certain regularity in the movement patterns so they are not completely random [gon08]. Only in [lau08] are mentioned advantages of NLP method for virtual environments. Here, a hybrid method, where a combination of a mouse motion prediction and NLP based on statistical approach is used to reduce latency. As the statistics are collected from zone to zone within a scene divided by regular grid, the information about continuous movement is lost.

In [bat02] a simple Markov model is used to estimate transition probabilities between adjacent cells based on movement history database. In [gam12] is used Markov chain of order m , which further increases prediction accuracy. Work [ash09] and [gam12] cluster GPS data into frequently visited locations (POI) [zho04]. Clustering to POIs is not suitable for DVW applications, because granularity of requested prediction for common

rendering algorithms is much higher. Mixed Markov-chain model [asa11] has been proposed to model behavior of individual pedestrians as well as a group of pedestrians with similar behavior. It uses combination of Markov chain and Hidden markov model (HMM) to construct the universal predictor. This approach has higher quantity compared to stand-alone Markov chain based methods, but the HMM method has high training complexity.

3 ENVIRONMENT DESCRIPTION

As each data transfer scheduling algorithm is closely related to used rendering algorithm and scene data representation, we will first briefly introduce our experimental framework.

3.1 Multi-resolution data representation

Multi-resolution data representation allows streaming and rendering of scene parts at coarser resolutions in the case of slow network connection, fast moving user or limited rendering capabilities of target devices. In our framework, the multi-resolution data is represented by three data layers including terrain geometry layer, terrain textures layer and cartographic layer (see Figure 1).

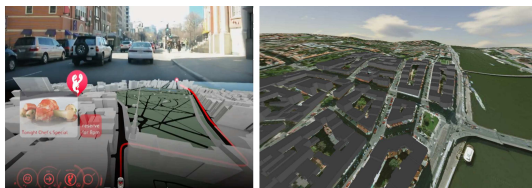


Figure 1: AIDA - 3D visualization and navigation system with augmented reality [aid13] (left), and our streaming and rendering system (right).

3.1.1 Terrain geometry and texture layer

The terrain layer contains height-map tiles (obtained from ASTER global digital elevation model [ast13]) which are further organized into an elevation data pyramid (see Figure 2).

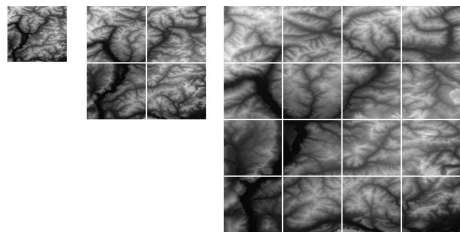


Figure 2: Each tile from coarser level (left) can be covered by its four children (center), continuing recursively (right) to the bottom of the pyramid. Each child tile covers one quarter of the area covered by its parent tile.

Each tile through all levels of the elevation data pyramid has resolution of 128×128 height points. The top

level of the pyramid contains single tile which covers the whole terrain at a much coarser resolution. The four child tiles cover the same area as their parent tile thus resulting in double resolution. Tiles at the bottom of the elevation data pyramid cover whole terrain at highest possible resolution.

The texture layer contains high resolution orthographic texture tiles which are mapped onto the terrain tiles. Each tile has resolution of 256×256 pixels and is also included inside a texture data pyramid (see Figure 3) similarly to the terrain tiles.



Figure 3: Part of the textures data pyramid which is created similarly like the elevation data pyramid.

As the elevation and textures data is obtained from real datasets, it is defined, that each elevation tile is covered by an array of 4×4 texture tiles. Consequently, each 128×128 terrain tile is covered by a texture data with resolution of 1024×1024 pixels.

3.1.2 Cartographic layer

The cartographic layer is created from Open Street Map (OSM) cartographic database, which contains definition of streets (geographic location, names, types, etc.), buildings (outlines, nested outlines, roof types, height, floor levels etc.) and other information. Each cartographic tile covers a single terrain tile at its finest resolution. No level of detail for the cartographic data is used. Download priority defined between the three layers is application dependent and is not the main point of interest of this paper.

3.2 Rendering algorithm

The scene is rendered using a set of concentric square rings around the user. Each ring is composed of a constant size grid of small patches. As the user moves, the patches which fall outside a render ring are asynchronously updated with new data from the three data layers at appropriate resolution (see Figure 4). To prepare and render each patch, only small parts of one or more terrain, texture and cartographic tiles are needed.

In our experimental evaluation we set that each ring is composed of a 12×12 grid of patches. Each patch covers an array of 13×13 elevation points. Consequently, each ring (whole square) needs at most 4 terrain tiles, 36 texture tiles and 4 cartographic tiles to prepare all its patches for rendering. As the user continuously

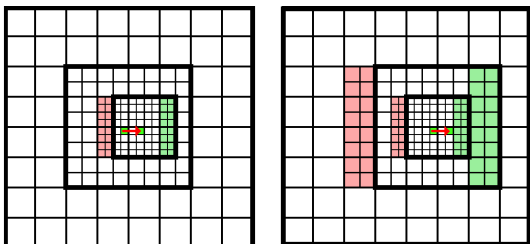


Figure 4: Example of ring patches update for three concentric rings. The red arrow symbol represent a moving user. The red patches have to be updated with the data covered by the green patches at particular resolutions.

moves, only subset of data tiles which cover the updated patches is needed.

Patches outside view frustum are not rendered but data tiles needed by these not visible patches are scheduled to be downloaded with low priority. This behavior is application dependent.

4 MOVEMENT DESCRIPTION

The movement of each user is defined as a continuous sequence of geographic coordinates (gps trajectory). Instead of working directly with gps trajectories, the NLP methods use mainly sequences of places of interest (places with high density of visiting users or places where the users stay for a long time etc.). These places can be discovered from input trajectories using various clustering mechanisms [ash03].

Once the clustering is finished, the input trajectories are encoded into sequences of visited places of interest (e.g. *home* → *work* → *shop* → *home*). Unfortunately, this approach is not very suitable for streaming of 3D virtual environments, because granularity of prediction based on the visited places of interests is usually too high to fit requirements of common streaming and rendering algorithms.

4.1 Trajectories projection

Instead of finding individual places of interest, we virtually divide the whole environment into a regular grid of square cells and project all the input GPS trajectories according to it so that each trajectory can be described as a continuous sequence of adjacent cells. The projection is repeated several times with a different resolution of the grid (see Figure 5).

The idea of the multiple resolutions of the grid is that particular render rings can be associated with selected resolution of the grid. It allows the NLP to be performed at particular resolution, thus allowing to compute priority or prefetch data tiles at appropriate resolution (see section 6).

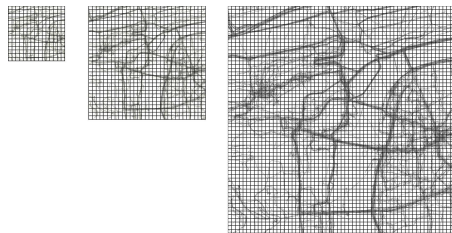


Figure 5: Subset of input trajectories projected to the regular square grid at various resolutions (increased by power of two from left to right).

4.2 Trajectories encoding

Once the projection is finished, each trajectory is encoded as a continuous sequence of adjacent cells (green cells in Figure 6) at selected resolution.

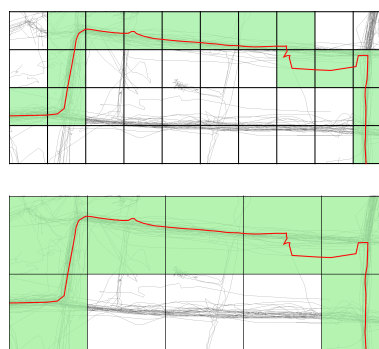


Figure 6: Example trajectory (red) projected at finest resolution (top) and one level coarser resolution (bottom).

Each projected trajectory is further encoded by chain code of eight directions [fre61] (see Figure 7) to efficient storage, and fast evaluation of prediction queries. For example, the red trajectory starting from left in Figure 6, will be encoded as a sequence of directions $2 \rightarrow 0 \rightarrow 0 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 4 \rightarrow 2 \rightarrow 2 \rightarrow 4 \rightarrow 4$ at the finer resolution and as a sequence $0 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 4$ at the coarser resolution.

7	0	1
6		2
5	4	3

Figure 7: Codes for eight possible movement directions from current (center) cell.

4.3 Trajectories storage

Trajectories encoded in the form of 8 directions are stored in a form which will be suitable for further described NLP methods. Assume that a whole trajectory is composed of a sequence of adjacent cells at selected resolution. We take successively each cell of the trajectory starting from the first one and determine specified

number of next successive movement directions from that cell along the trajectory. Next 9 successive directions from each cell are used (see section 7). Let's assume that each direction can be encoded using 3 bits. Then a sequence of 9 directions can be stored using 27 bits plus 4 bits to encode its length, because sequences of last 9 cells can be shorter. Therefore, each sequence of directions can be stored using single 32 bit integer value. This value forms a local movement pattern defined for each cell of a trajectory. These local movement patterns are computed for each cell for all input trajectories projected at all resolution levels. The local movement patterns form a knowledge database which is used as an input for the following NLP methods.

5 NEXT LOCATION PREDICTION

NLP methods suitable for our case have to satisfy several requirements: efficient learning, fast adaptation to new behavior of each individual user, high prediction accuracy and quantity, and fast evaluation of prediction queries. The most important requirement is prediction accuracy, because each wrong prediction decreases data transfer efficiency and rendered scene quality.

Learning NLP methods can be considered efficient, if the knowledge database can be modified by adding or removing trajectories so the prediction methods are not forced to be completely relearned. Considering fundamental characteristics of NLP methods [pet06], this requirement is met by both the Markov chain based predictor [ash03], [pet06], [gam12] and also by the K-state predictor [pet03].

5.1 Markov chain based predictor

The Markov chain (MCH) based predictor is based on the definition of Markov chain of order j . A Markov chain of order j selects its next state depending upon j past states. In our case, its state space is defined by the eight movement directions. Each Markov chain operates with transition frequency value which counts an overall number of applications of corresponding movement direction after a sequence of j past movement directions (see Figure 8).

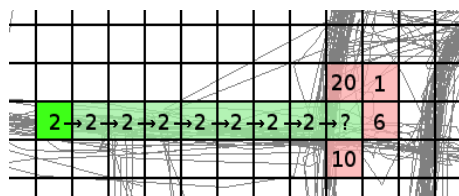


Figure 8: Local movement pattern with length of nine directions related to the left green cell.

The local movement pattern shown in Figure 8 moves the user from the green cell to current cell marked as "?" by application of 8 right transitions (2 →). The

9th transition from the pattern moves the user to one of the red cells. The red cells contain transition frequency counters incremented by trajectories with the same local movement pattern related to the leftmost green cell.

The cell with highest value of the frequency counter can be selected as a prediction result. Confidence of such prediction is computed as the ratio between the value of selected frequency counter and the sum of all frequency values adjacent to cell "?". If confidence of the prediction is less than 90%, it will be marked as not confident. The lower the confidence threshold, the lower the accuracy. If the confidence computation is skipped, the prediction accuracy is decreased by 5-8% depending on prediction resolution level. We decided to set the confidence threshold to 90% because the prediction accuracy is the most important parameter for us. If the confidence is for example 50%, i.e. two directions should be considered as a prediction result, then it is not reasonable to apply the prediction result.

MCH based predictor has a property that is cannot fast adapt to changes in habits [pet06] neither temporary behavior changes (street closures because of road works etc.) of individual users. Consequently, this characteristic can lead to a confident but a wrong prediction for quite long time until the frequency counters reflect some change.

5.2 K-state predictor

The concept of k-state predictor (KSP) as a NLP method inside a smart office building has been introduced in [pet03]. It can be also successfully used with the trajectories encoded by chain code of eight directions.

The KSP is constructed as a simple finite automaton with k-states. Currently, we use only a 2-state predictor (2SP) with eight contexts where each context represents one direction. The two states are a weak and a strong state (see Figure 9).

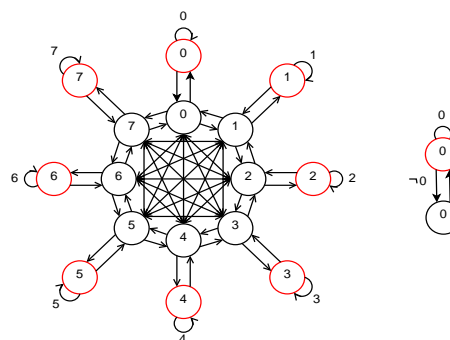


Figure 9: The 2SP with eight strong (red) and eight weak states. Each context of the predictor represent one direction from the used chain code of eight directions.

In case the 2SP is in the strong state then the appropriate movement direction is returned as prediction re-

sult, otherwise no prediction is returned. Instead of increment frequency counters, the 2SP switches between its contexts and states. Each 2SP starts with undefined context. After applying e.g. eight right directions from the local movement pattern shown in Figure 8 followed by single up direction, it will switch to up context at the weak state. If the same pattern is repeated followed by the up direction again, the up context will switch to the strong state and the up direction will be returned as a prediction result. This functionality can be easily extended by other dimensions like time, day of a week etc.

5.3 Proposed prediction scheme

Input of the prediction scheme is a current movement pattern. It is described by a reference cell and sequence of 8 movement directions. Current cell of the user can be determined by applying these 8 directions from the reference cell. All local movement patterns (stored at client) related to the input reference cell are loaded and only the patterns which match the current movement pattern are selected. As the current input movement pattern has a length of 8 directions and the stored movement patterns have a length of 9 directions, the 9th direction can be used to construct the 2SP.

If the 2SP does not return any prediction, the MCH based predictor will be used. As the MCH based predictor needs the knowledge database of all users in the system, it will be performed and evaluated at server. The 9th direction from all the matching local movement patterns (stored at server) related to the input reference cell are used to increment the frequency counters.

Prediction quantity of both predictors is low because of lack of movement data for new users, users with new behavior or low confidence of the predicted directions. The standard solution to this problem is prediction by partial matching (PPM). We further extend it with application of recursive motion function.

Prediction by partial matching:

Prediction by partial matching (PPM) is based on shortening length j of the local movement patterns so that it successively moves the reference cell toward the current cell until a prediction succeeds or other conditions are reached. The disadvantage of this approach is that the shorter the local movement pattern is, the less prediction accuracy there is. We determine minimum acceptable local movement pattern length so the next location predictors have increased quantity, but still high prediction accuracy. If a predicted cell is based on a local movement pattern shorter than the observed minimal length, we mark such prediction as not confident. Even if such a prediction is not confident, it is still more accurate than motion function based predictors (see section 7). Therefore, such a not

confident prediction is used only to compute priorities of missing scene parts, but not for prefetching them.

Recursive motion function:

In case both the Markov chain and 2-state prediction methods did not respond to a prediction query, the proposed prediction scheme predicts next movement using a state-of-the-art recursive motion function (RMF). The input of the RMF method is a sequence of past GPS coordinates so that it can predict next position based on motion function determined from these past positions. The predicted position is clustered using the grid-clustering method only at the finest resolution. In case the cell determined from the predicted position does not differ from the current cell, the RMF predictor is applied again to predict more steps ahead until a difference between the current cell and the predicted cell appears. The RMF method has lower accuracy but high quantity compared to the NLP methods. Therefore, we use the RMF method result only to compute priorities of missing scene parts, but not for data prefetching.

6 DATA TRANSFER SCHEDULING

The main goal of scheduling of data transfer for our DVW application is to achieve effective data transfer with maximum rendering quality during scene exploration. We use a hybrid client-server communication approach, where both client and server can prefetch or request missing scene parts.

6.1 Rendering requirements

Rendering algorithm described in section 3.2 exactly determines data tiles which are needed to render the current view. As the user continuously moves through the scene, the rendering algorithm generates requests to download tiles which are not available in cache memory. The scheduling algorithm first requests tiles for coarser resolution rings, continuing to the finer resolutions. Therefore, the correspondence between the rendered rings and the necessary tiles for the three data layers are determined by outer boundary of each ring.

6.2 Application of the prediction scheme

The scheduling algorithm returns predicted cell which can be used to compute download priority of missing data tiles needed for rendering the current view or to prefetch data tiles needed for rendering the future views.

6.3 Download priority determination

For a fast moving user or slow network connection it is not possible to transfer all requested tiles on time. Even worse, some tiles required for rendering the current view will be downloaded late, so they are no longer needed for rendering the current view.

Priority of all missing data tiles within the given render ring is computed based on a cell C . The cell C is computed as $C = C_c + k * (C_p - C_c)$, where C_c is the center cell of given render ring, C_p is predicted cell and k is a constant which translates the cell C outside the given render ring boundaries. The priority of all missing data tiles is computed as a distance between each missing data tile from the cell C . The less the distance is, the higher the priority.

6.4 Data prefetching

Let's assume that current location of a user is determined by the center cell C_c at selected resolution. The predicted cell C_p is always adjacent to the cell C_c . Every time, the proposed prediction scheme returns confident prediction, the predicted cell C_p is used to identify render rings borders for that cell C_p . These borders specify all the required data tiles if the user will follow the predicted cell C_p . Actually we map one cell from the projection grid to one texture tile from the texture layer at appropriate resolution (see Figure 10). The same principle is applied to the terrain and cartographic layers.

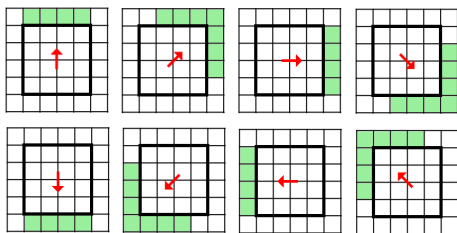


Figure 10: Example one-by-one mapping between tiles from the texture layer and cells of the projection grid. The red arrows sign the predicted direction and the green cells are prefetched data tiles. The black squares are the render rings for the current cell C_c .

Tiles determined by the prefetching algorithm can be scheduled to be downloaded only if no tiles are missing by the current render rings. We select this strategy as we need to achieve both scene quality and data transfer efficiency.

7 EXPERIMENTAL EVALUATION

A proof-of-concept client-server framework which runs on iPad, renders the described virtual environment and exploits the proposed scheduling algorithm was implemented. Prediction accuracy and quantity of the used prediction methods are evaluated as well as render quality and data transfer efficiency during walkthrough the environment.

7.1 Input trajectories dataset

All the experiments were done with trajectories obtained from Open Street Map gpx database from rectangle area specified by two $[latitude, longitude]$

corners as $min = [49.9812545, 14.230042]$ and $max = [50.182172, 14.617306]$ which covers a large city. The dataset contains 2,648 gpx trajectories recorded by various users as continuous sequences of $[latitude, longitude]$ coordinates. The input trajectories are projected to a regular grid at resolution starting from $[0.000278, 0.000278]$ degrees per cell and further increased by the power of two, finishing with five resolution levels.

Accuracy and quantity of the NLP methods are evaluated using 20-fold cross validation (each trajectory set contains 132 trajectories). The validation is performed so that 19 sets are used for learning both the MCH based predictor and also the 2SP. The remaining set is always used to evaluate prediction accuracy and quantity. We repeat this process 20 times for different testing sets and compute the resulting accuracy and quantity by averaging the particular results.

As the trajectories can be obtained only as anonymous records, we cannot assign them to individual users. Instead, we assign all the input trajectories to a single user and evaluate both the MCH based predictor and the 2SP with this assumption. Practically, the quantity of the 2SP will be less than the quantity of MCH based prediction, especially for new users.

7.2 Prediction accuracy and quantity

We decided to store 9 successive directions for each local movement pattern. Then 8 directions can be used to match the current movement pattern with the stored local movement patterns. Experiments with the input trajectories show that 9 directions are sufficient, because the change of accuracy, when the length of the matched sequence of directions is longer than 6 is small (see Figure 11).

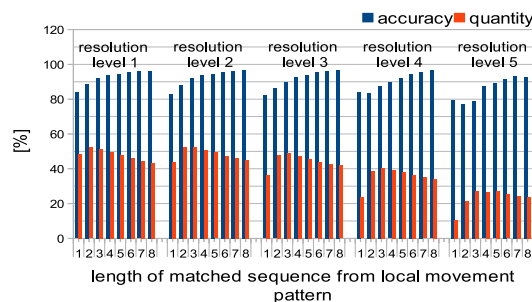


Figure 11: Comparison of accuracy and quantity of MCH predictor for different length of matched sequence of directions and all 5 resolutions of the projection grid.

The prediction quantity is highest for matched sequences of directions with length from two to three direction over all resolution levels as is shown in Figure 11. The quantity is lower for shorter lengths

because the predictions are often marked as not confident (confidence threshold is selected at 90%) and also for longer length, because less matched patterns were found. As a compromise between accuracy and quantity we set length of the matched sequence of directions to 6 directions. The accuracy of both the 2SP and MCH based predictor is higher than 90% for all resolutions of the projection grid (see Figure 12).

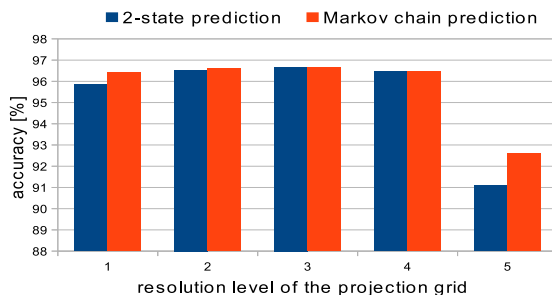


Figure 12: Comparison of prediction accuracy between 2-state (2SP) and Markov chain (MCH) based prediction.

The proposed prediction scheme uses also prediction by partial matching for both MCH based and 2-state predictors. Figure 13 shows effect of PPM to accuracy and quantity of the MCH based predictor for all PPM orders at all resolutions of the projection grid. The PPM order is the maximum allowed length of shortening a sequence of matched directions from the local movement patterns.

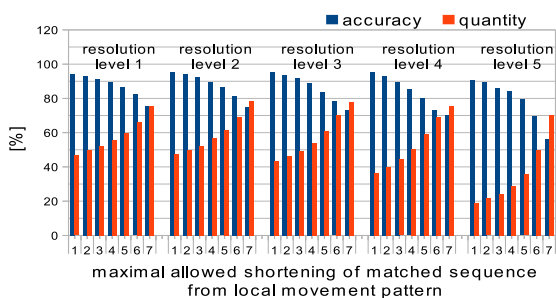


Figure 13: Effect of PPM optimization to accuracy and quantity of MCH based prediction. Initial length of matching sequence is 8 directions.

The results show that the higher the allowed shortening is, the less the accuracy is, but with increased quantity. Based on the PPM results, we have decided to allow the shortening at most by two directions, otherwise the prediction is marked as not confident and is rather used to compute only priorities of current missing data tiles.

Even though prediction accuracy of the highest PPM order at the first resolution level is relatively low, it is still higher than prediction accuracy of the state of the art recursive motion function used as the next location predictor. The prediction accuracy of the recursive mo-

tion function is 55% and quantity is 97%. We use 6 past GPS locations to create the RMF predictor and we use it only for prediction at the finest resolution of the projection grid. The accuracy of simple linear predictor is 39% and its quantity is 99%.

7.3 Scheduling scheme evaluation

The goal of the proposed scheduling scheme is to keep both scene quality and data transfer efficiency as high as possible. We measure the scene quality as the ratio between the time a data tile is available for rendering and time the data tile is needed for rendering. The result is computed as weighted average over all tiles for particular levels. Data transfer efficiency is defined as the ratio between amount of downloaded tiles and how these tiles contribute to rendered scene quality.

We changed speed of the user, network latency and connection bandwidth and measured scene quality and data transfer efficiency with the proposed prediction scheme. The experiments are performed on local area network with network latency and bandwidth emulated at linux server using "tc" commands with "netem" kernel component. We use the first set of input trajectories to perform three experiments.

Average data transmission speed on current 3G networks vary from approx. 1000kbps to 3000kbps and latency from approx. 50ms to 100ms [tmc11], but it depends on many conditions. Therefore, in the first experiment (see Figure 14) we set motion speed to 60km/h, bandwidth to 2000kbps and latency to 100ms.

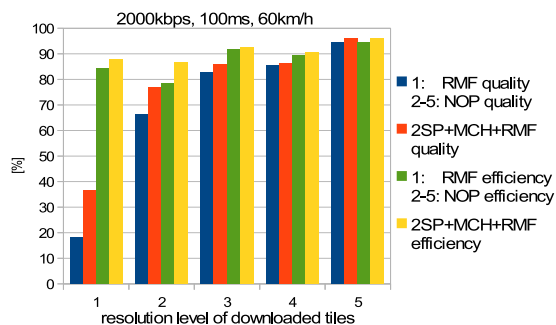


Figure 14: Comparison of quality and data transfer efficiency of the scheduling scheme and RMF (used for the first finest resolution) and no prediction (NOP) for the other resolutions.

The results of the first experiment show that for high bandwidth and relatively high latency for this kind of application the proposed prediction scheme outperforms the RMF used for the finest resolution and no prediction used for the coarser resolutions. Except the rendering quality at the finest resolution, the results are similar, because the network bandwidth is high so all tiles at all resolutions are downloaded during rendering.

In the second experiment we change speed to 130km/h. The results show (see Figure 15) both increased quality and also significantly increased data transfer efficiency with the proposed prediction scheme at second and third resolution level. It means the downloaded tiles are needed for rendering for longer time compared to the RMF and no prediction approaches. The finest resolution of data tiles was not downloaded at all because the speed is too high for the selected bandwidth and latency.

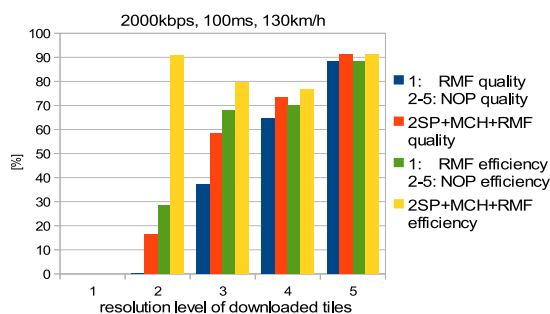


Figure 15: Comparison of scene quality and data transfer efficiency between proposed scheduling scheme and RMF (used for the first finest resolution) and no prediction (NOP) for the other resolutions.

In the last experiment, we set high speed, low bandwidth at 200kbps and zero latency just to show the effect of prediction at low bandwidth networks for fast moving user (see Figure 16).

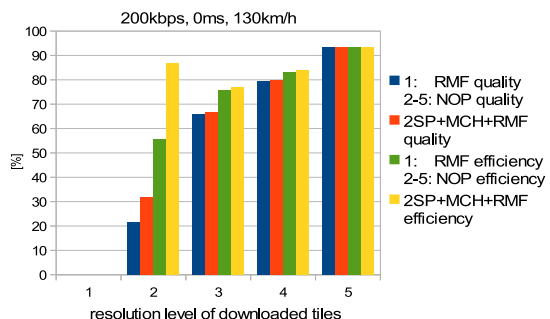


Figure 16: Comparison of scene quality and data transfer efficiency between proposed scheduling scheme and RMF (used for the first finest resolution) and no prediction (NOP) for the other resolutions.

The results show that the proposed scheduling scheme significantly increases data transfer efficiency and scene quality at the second resolution level.

8 CONCLUSION

The proposed scheduling scheme outperforms the RMF at the finest resolution level in both the scene quality and data transfer efficiency. At coarser levels the motion functions cannot be used because of their low accuracy. The effect of the proposed prediction scheme is less significant at the coarser levels, because the amount

of time for which the coarser tiles are used for rendering is longer compared to the higher resolution tiles.

For applications, where the transfer efficiency is sufficiently high, it will be possible to predict more than one next location ahead. It will increase scene quality for slow moving users or at high speed networks. With more steps ahead, the possibility that the downloaded tiles will not be used for rendering grows up especially in the case of wrong prediction.

9 ACKNOWLEDGMENTS

This work has been funded by the Technological Agency of Centrum kompetence ve zpracovani vizualnich informaci (V3C - Visual Computing Competence Center, TACR, TE01020415). Special thanks also go to Cadwork Informatik CI AG and Cadwork development team in Brno for supporting of this work.

10 REFERENCES

- [aid13] Volkswagen of America, Massachusetts Institute of Technology (SENSEable City Lab and Personal Robots Group of Media Lab). AIDA - Affective, Intelligent Driving Agent. <http://senseable.mit.edu/aida/>.
- [ali10] Ali, M. E.; Tanin, E.; Zhang, R.; aj.: A motion-aware approach for efficient evaluation of continuous queries on 3D object databases. *The VLDB Journal*, Vol. 19, No. 5, 2010, pp.603–632, ISSN 1066-8888.
- [asa11] Asahara, A.; Maruyama, K.; Sato, A.; aj.: Pedestrian-movement prediction based on mixed Markov-chain model. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS '11*, New York, NY, USA: ACM, 2011, ISBN 978-1-4503-1031-4, pp.25–33.
- [ash03] Ashbrook, D.; Starner, T.: Using GPS to learn significant locations and predict movement across multiple users. *Personal Ubiquitous Comput.*, Vol.7, No.5, pp.275–286, 2003, ISSN 1617-4909.
- [ast13] Jet Propulsion Laboratory: ASTER Global Digital Elevation Map. <http://asterweb.jpl.nasa.gov/gdem.asp>.
- [ash09] Ashbrook, D.; Starner, T.: Using GPS to learn significant locations and predict movement across multiple users. *Personal Ubiquitous Comput.*, Vol.7, No. 5, 2003, pp.275–286, ISSN 1617-4909.
- [bat02] Bhattacharya, A.; Das, S. K.: LeZi-update: an information-theoretic framework for personal mobility tracking in PCS networks. *Wirel. Netw.*, Vol.8, No.2/3, 2002, pp.121–135, ISSN 1022-0038.
- [chi98] Chim, J. H. P.; Green, M.; Lau, R. W. H.; et al.: On caching and prefetching of virtual objects

- in distributed virtual environments. In *Proceedings of the sixth ACM international conference on Multimedia*, MULTIMEDIA '98, New York, NY, USA: ACM, 1998, ISBN 0-201-30990-4, pp.171–180.
- [chi03] Chim, J.; Lau, R. W. H.; Leong, H. V.; aj.: CyberWalk: A Web-Based Distributed Virtual Walkthrough environment. *IEEE Trans. on Multimedia*, Vol.5, 2003: pp.503–515.
- [fal93] Falby, J.; Zyda, M.; Pratt, D. R.; aj.: NPSNET: Hierarchical data structures for real-time three-dimensional visual simulation. *Computers & Graphics*, Vol.17, No.1, pp.65–69, 1993.
- [fre61] Freeman, H.: On the encoding of arbitrary geometric configurations. In *IEEE Trans. Electron. Comput.*, pp.260-268, 1961.
- [gam12] Gambs, S., Killijian, M.-O., del Prado Cortez, M. N. n.: Next place prediction using mobility Markov chains. In *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, MPM '12, New York, NY, USA: ACM, 2012, ISBN 978-1-4503-1163-2, pp.3:1–3:6.
- [gon08] González, M. C.; R., C. A. H.; Barabási, A.-L.: Understanding individual human mobility patterns. *CoRR*, Nature 453, pp.779-782, 2008.
- [hes98] Hesina, G.; Schmalstieg, D.: A Network Architecture for Remote Rendering. In *Proceedings of the Second International Workshop on Distributed Interactive Simulation and Real-Time Applications*, DIS-RT '98, Washington, DC, USA: IEEE Computer Society, 1998, ISBN 0-8186-8594-8, p.88.
- [lau08] Lau, R. W.; Chan, A.: Motion in Games. Chapter on Motion Prediction for Online Gaming, Berlin, Heidelberg: Springer-Verlag, 2008, ISBN 978-3-540-89219-9, pp.104–114.
- [li04] Li, F. W. B.; Lau, R. W. H.; Kilis, D.: GameOD: an internet based game-on-demand framework. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '04, New York, NY, USA: ACM, 2004, ISBN 1-58113-907-1, pp.129–136.
- [mar11] Marvie, J.-E.; Gautron, P.; Lecocq, P.; et al.: Streaming and synchronization of multi-user worlds through HTTP/1.1. In *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, New York, NY, USA: ACM, 2011, ISBN 978-1-4503-0774-1, pp.111–120.
- [osm13] Open Street Map. Open Street Map Wiki - Buildings. <http://wiki.openstreetmap.org/wiki/Building>.
- [pet06] Petzold, J., Bagci, F., Trumler, W., and Ungerer, T.: Comparison of different methods for next location prediction. In *LNCS, Mobile and Ubiquitous Computing*, pp.909-918, 2006.
- [pet03] Petzold, J., Bagci, F., et al.: Global and Local State Context Prediction. *Artificial Intelligence in Mobile Systems 2003 (AIMS 2003) in Conjunction with the Fifth International Conference on Ubiquitous Computing*, Seattle, USA, 2003.
- [sch96] Schmalstieg, D.; Gervautz, M.: Demand-Driven Geometry Transmission for Distributed Virtual Environments. *Comput. Graph. Forum*, Vol.15, No. 3, 1996: pp.421–431.
- [sch06] Schneider, J.; Westermann, R.: GPU-friendly high-quality terrain rendering. In *The 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2006 (WSCG 2006)*, Bory, Czech Republic, 2006.
- [tao04] Tao, Y.; Faloutsos, C.; Papadias, D.; aj.: Prediction and indexing of moving objects with unknown motion patterns. In *Proceedings of the 2004 ACM SIGMOD int. conference on Management of data*, SIGMOD '04, New York, NY, USA: ACM, 2004, ISBN 1-58113-859-8, pp.611–622.
- [tel01] Teler, E.; Lischinski, D.: Streaming of Complex 3D Scenes for Remote Walkthroughs. *Computer Graphics Forum*, Vol. 20, No. 3, 2001, pp.17–25, ISSN 0167-7055.
- [tmc11] T-mobile Czech Republic, P3 Communications and BUT: Press release - 3G and 3G networks performance testing results. http://tpress.cz/tiskove_zpravy/2011/1224/.
- [wan09] Wang, W.; Jia, J.: An incremental SMLAOI algorithm for progressive downloading large scale WebVR scenes. In *Proceedings of the 14th International Conference on 3D Web Technology*, Web3D '09, New York, NY, USA: ACM, 2009, ISBN 978-1-60558-432-4, pp.55–60.
- [zho04] Zhou, C.; Frankowski, D.; Ludford, P.; aj.: Discovering personal gazetteers: an interactive clustering approach. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, GIS '04, New York, NY, USA: ACM, 2004, ISBN 1-58113-979-9, pp.266–273.

Metrics of human perception of vanishing points in perspective sketches

Raquel Plumed

Department of Mechanical
Engineering and Construction
Universitat Jaume I
Avda. Sos Baynat, s/n
Spain (E-12080),
Castellón de la Plana
plumed@uji.es

Pedro Company

Ins. of New Imaging Technology,
Universitat Jaume I
Avda. Sos Baynat, s/n
Spain (E-12080),
Castellón de la Plana
pcompany@uji.es

Peter A.C. Varley

Ins. of New Imaging Technology,
Universitat Jaume I
Avda. Sos Baynat, s/n
Spain (E-12080),
Castellón de la Plana
varley@emc.uji.es

ABSTRACT

This paper describes an experiment aimed at discovering how humans perceive vanishing points depicted in perspective sketches of engineering shapes. The goal is to find criteria and metrics for an algorithmic approach to replicate human perception of vanishing points. A new approach is required for Sketch-Based Modelling, since most current image analysis approaches take 2D camera images as their input, so do not solve satisfactorily the problem of geometrical imperfections inherent in sketches.

We have conducted a pilot experiment to determine which vanishing points are perceived by people, and under what circumstances they are perceived. We test the hypotheses that (i) people are able to detect and locate vanishing points in sketches in spite of their inherent imperfections, and (ii) factors such as distance of vanishing points from the sketch and number and lengths of lines converging at the vanishing points influence their perception.

Keywords

Sketch-Based Modelling, Perspective projection, Vanishing points.

1. INTRODUCTION

Our goal is to assist designers to interact in a friendly way with computers. The advantages of this are well-documented [Joh09], but we believe that this interaction must not come at the cost of unexpected behavior. People will only trust the computer if they feel that it interprets things more or less as they do. To this end, we intend to develop algorithmic approaches which replicate human perception when reconstructing models depicted in perspective sketches of engineering designs.

Hence, we should know how humans interpret design sketches. Here, we describe an experiment aimed at discovering: which vanishing points people perceive, where they are located; and what geometrical flexibility in their locations can be tolerated.

After analysing the results, we obtain criteria and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

metrics which will help to create algorithms which mimic human behaviour.

We first revisit the background of central projection and vanishing points. We then describe the design of our experiment and analyse our results.

2. DEFINITION OF TERMS

In the field of sketch-based geometric reconstruction, one approach is to tackle the issue as an artificial perception problem [Lip96], [Var03], [Com04], [Yua08], [Tia09]. Human beings have an intrinsic or learnt capability to mentally reconstruct three-dimensional objects from 2D images by means of pictorial clues [Gol99], [Hof00]. Here, we are interested in a specific pictorial clue, the vanishing point (VP).

In *perspective* projection, parallel lines not parallel to the image plane converge to a *vanishing point* (VP). The fundamentals of perspective were first codified in Durer's Four Books on Measurement in 1522, and their effects on how we see and draw are well-known [Pal99], [Wri83].

The number of VPs in an image depends on the orientation of the depicted object relative to the projection plane. A normalon polyhedron (one with all its edges parallel to one of the three main

Cartesian axes) may produce three distinct situations: a) two axes parallel to the image plane and just one VP where the lines parallel to the third axis converge; b) one axis parallel to the image plane and two VPs where the lines parallel to the other two axes converge, and c) no axis parallel to the image plane and three VPs where the lines parallel to the three axes converge. The three varieties, known as one, two and three vanishing point perspectives, are shown in Figure 1.

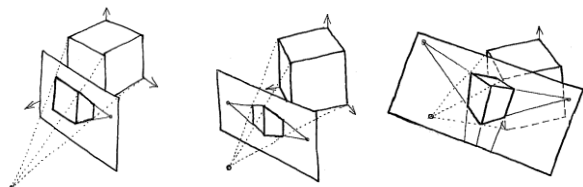


Figure 1. Linear perspective with one (left), two (middle) and three (right) vanishing points.

Another important distinction is between *main* and *oblique* VP. For a general polyhedron with n different sets of parallel edges, the varieties of linear perspective become $n, n-1, n-2, \dots$, depending on the number of groups of parallel edges in the model which are parallel to the image plane. Figure 2 shows an example with only two main VP: the vertical axis is parallel to the image plane, so vertical lines do not converge; however, an additional oblique VP results from the convergence of the lateral edges of the wedge.

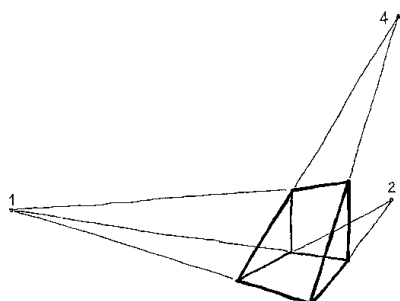


Figure 2. Wedge with two main vanishing points plus one oblique vanishing point.

There is no theoretical distinction between different locations of VPs relative to the object. But there is a useful practical distinction between VPs located *inside* and *outside* the object (Figure 3). Internal VPs are typical in indoor architectural scenes, but are rarely used to depict engineering products. Hence, we do not study them here.

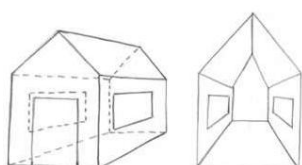


Figure 3. Linear perspectives of a prismatic shape with one external (left) and internal (right) vanishing point.

3. HYPOTHESES

The purpose of our experiment is to obtain criteria and metrics for algorithms which mimic human perception in detecting vanishing points in a sketch. Here, we propose the hypotheses to be tested:

1. Human beings perceive the existence of intended vanishing points in sketches of 3D polyhedral shapes, in spite of their inherent imperfections.
2. Humans beings are able to locate quite precisely those vanishing points which are neither too close to nor too far away from the drawing.
3. The lengths of lines influence convergence detection. The longer the lines, the easier it is to detect a vanishing point.

4. Design of the experiment

We designed our pilot experiment as follows. First, we selected a set of sketches. Then, we asked a group of subjects to determine the approximate number and location of vanishing points implied by a sketch, and also to label the different sets of parallel edges. Finally, we analysed the results to determine to what extent people agree in perceiving the same vanishing points, and what are the most influential factors in this perception process.

1		2	
3		4	
5		6	
7		8	
9		10	
11		12	
13		14	
15		16	
17		18	

Figure 4. Set of sketches used in the experiment

Here, we have not considered the scale as a main factor, because we guess that it only affects human perception for too small or too big drawings, where

some lines may be perceived with difficulty. But this is not the case for design drawings sketched on pen input devices.

Set of sketches

Our set of test sketches is derived from typical Engineering Design training exercises (Figure 4).

The sketches were selected to meet the following criteria:

- They should be simple, containing no unnecessary features or details which could divert the attention
- They should be representative of shapes usually sketched in engineering design processes
- They should represent polyhedral shapes, in both natural and wireframe styles.
- They should be tidied line drawings in central projection style.
- They should represent different varieties of central projection (one, two, three vanishing points).
- Some of them should contain vanishing points corresponding to oblique directions.

The result was the set of 18 sketches shown in Figure 4.

We then circulated these sketches to our test subjects, who marked them up as requested.

Finding vanishing points

The first task for the test subjects was to find and mark all the VPs for a given sketch. We gave them standardised A4 questionnaire containing a short explanation of the task, a visual example (Figure 5), and two sketches selected randomly from the test set.

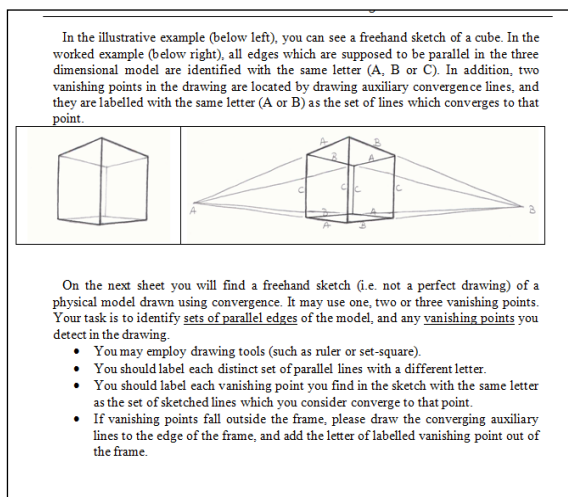


Figure 5. Questionnaire.

Finding non-convergent groups of lines

We also wanted to know why possible VPs were left unmarked: were they dubious, or did they

corresponding to perceived parallel (i.e. non-converging) edges? Hence, as second task, the subjects were asked to mark all those groups of lines representing parallel edges, and label them as separate sets.

Participants

The bulk of the subjects who participated in the experiment were drawn from diverse departments of the same university, and included mechanical, electric and industrial engineers, architects, designers and artists. The level of experience ranged from undergraduate students to professors. Of the 149 participants, 92 (61.7%) were engineers, 23 (15.4%) architects, 20 (13.4%) had artistic knowledge, and 14 (9.4%) were school-age (17-18 years) students whose studies included technical drawing.

We found no systematic differences in the results between subjects from different backgrounds. In the analysis below, we treat the subjects as a single homogeneous group.

5. Results

We issued 298 questionnaires, of which 291 were returned. At this stage, we removed from the study those questionnaires which did not give coherent results. The most common mistakes were due: (i) to misunderstanding the concept of vanishing points, marking erroneous VPs which did not correspond to intersections of lines of the drawing (Figure 6a); or (ii) failing to understand the drawing as a representation of a 3D shape with parallel edges, marking erroneous VPs at the intersection of lines of the drawing which did not represent parallel edges of the depicted 3D shape (Figure 6b).

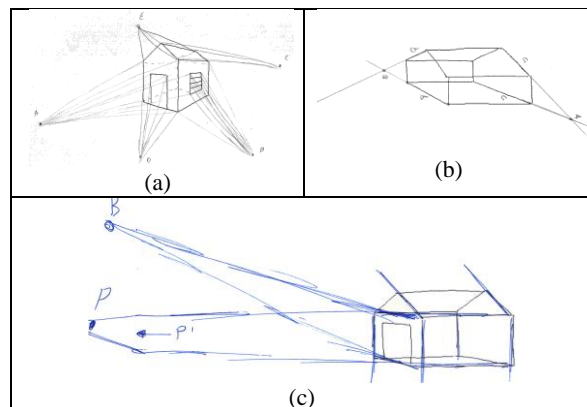


Figure 6. Erroneous vanishing points.

Some other questionnaires were also dismissed or only partially considered as it was difficult to interpret them objectively. Most of these dubious questionnaires were of sketch 15 (Figure 6c).

From the total of 291 collected questionnaires, we were left with 266 coherent responses. Of these, 7 were explicitly marked as not containing any VPs

(four of these were sketch 18, two were sketch 15 and one was sketch 11).

Qualitative validation of the first hypothesis

We grouped the lines of each sketch by the parallel edges they belong to in the 3D object. Since all of the sketches depict polyhedral models, they always have at least three main axes; we labelled these three main axes as X, Y and Z (or 1st, 2nd and 3rd), as in Figure 7; verticality is always labelled as Z (axis 3). Objects 3, 7, 10, 11, 12 and 15 include one or two additional oblique axes, labelled as axes 4 and 5.

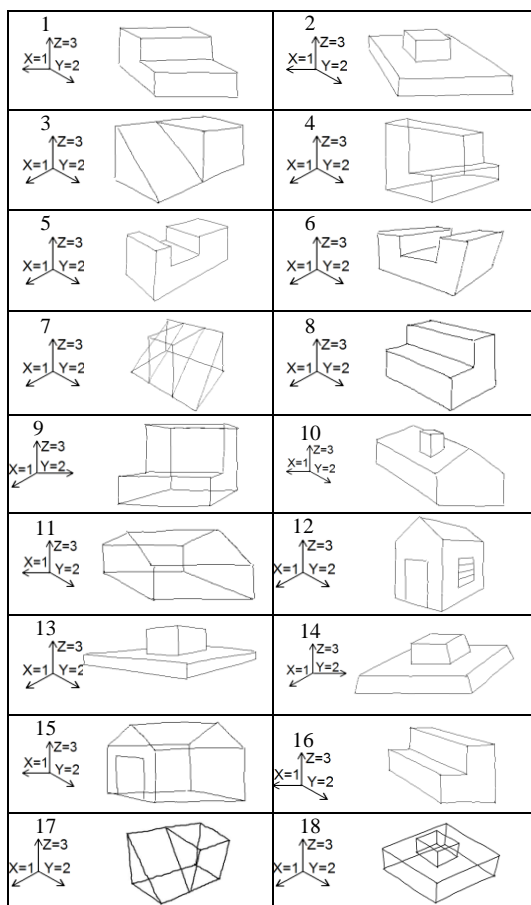


Figure 7. Main axis in the set of sketches used in the experiment

We then analysed the perception of VPs for each sketch and each candidate axis. Table 1 lists the percentage of polled people who perceived convergence for each sketch and axis.

These results show that people perceive the existence of a VP for axis 1 in sketches 3, 4, 5, 6, 9, 12, 13, 14 and 17. The existence of VP 1 is uncertain for sketches 7 and 8. Lines parallel to axis 1 are perceived as non-convergent for sketches 1, 2, 10, 11, 15, 16 and 18.

Similarly, people perceive the existence of a VP for axis 2 in all sketches except 3 and 15 (where the

vanishing point is uncertain), and 9, 14, 17 and 18, which are perceived as non-convergent.

Sketch	Axis 1 (X)	Axis 2 (Y)	Axis 3 (Z)	Axis 4	Axis 5
1	46.67	93.33	33.33		
2	17.65	94.12	23.53		
3	100.00	72.22	5.56	66.67	
4	100.00	76.92	0.00		
5	100.00	84.21	5.26		
6	100.00	85.71	78.57		
7	68.75	93.75	6.25	75.00	
8	72.22	94.44	0.00		
9	100.00	7.14	0.00		
10	7.14	92.86	0.00	71.43	
11	22.22	77.78	0.00	44.44	33.33
12	100.00	76.47	5.88	35.29	5.88
13	100.00	92.31	0.00		
14	83.33	8.33	75.00		
15	28.57	57.14	28.57	28.57	42.86
16	6.67	100.00	0.00		
17	94.74	31.58	84.21	68.42	
18	6.25	12.5	68.75		

Table I. Perceived vanishing points

Axis 3 is perceived as convergent in sketches 6, 14, 17 and, with less certainty, in sketch 18. No-one perceived convergence in sketches 4, 8, 9, 10, 11, 13 and 16. It appears that engineering designers are less used to sketching convergence in the vertical direction, and subjects seem to be less willing to perceive convergence for this axis.

Convergence of oblique lines is only perceived with certainty for sketch 7; it is uncertain for sketches 3, 10 and 17. It appears that (i) humans do not readily perceive oblique convergence, but (ii) a large number of lines (as in sketch 7) help humans to identify convergence to an oblique VP.

In summary, convergence of the main axes seems to be readily and generally perceived, regardless of sketching imperfections. It is somewhat more difficult to perceive VPs for oblique axes. In Section 6, we shall attempt to determine the minimum threshold of angle of convergence and number of lines which guarantee a general perception of VPs.

Qualitative validation of the second hypothesis

For each sketch, we superimposed all VPs located by the subjects. For example, the red points in Figure 8 are the locations of the VP of lines aligned with axis 1, the blue points those aligned with axis 2, and the green points those aligned with axis 4. The results clearly show that people agree about the orientation angle along which the VP is located, but fail to agree about the position of the VP along this line.

It also appears that clouds of clearly-perceived VPs are shorter (and may be bounded by an ellipse), while clouds of uncertain VPs tend to be longer and resemble a straight line.

It also appears that the dispersion in the location increases when a) the VP is distant from the drawing, and b) when the group of lines is small and/or the lines are short.

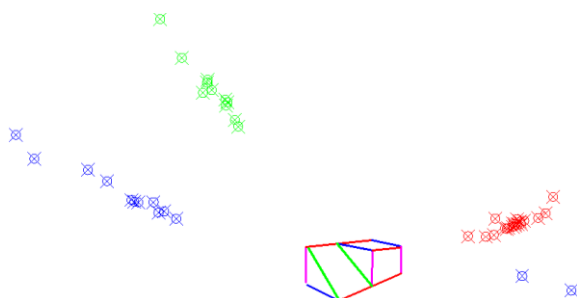


Figure 8. Superimposition of all VPs located by the subjects for sketch 3.

We note in passing that in 4 of the questionnaires where objects are depicted by natural drawings, subjects drew the hidden lines in order to find the VP locations. This demonstrates that they knew how to interpret the sketches, but felt unable to fix VP location with precision. Figure 9 (sketch 12) shows how one subject even used hidden lines to locate an additional VP which corresponds just to a single line of the original sketch.

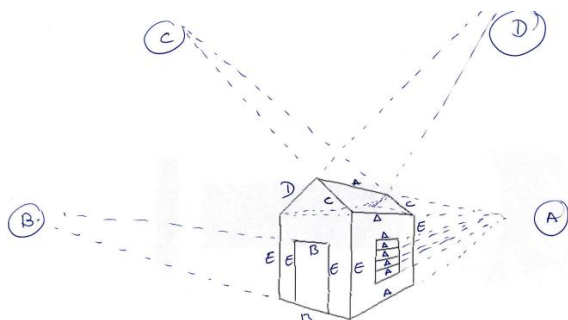


Figure 9. Hidden lines for VPs location in sketch 12.

Analysing the questionnaires, we noticed that the subjects used one of three strategies to overcome the imperfections of the sketches and find the most likely location for VPs.

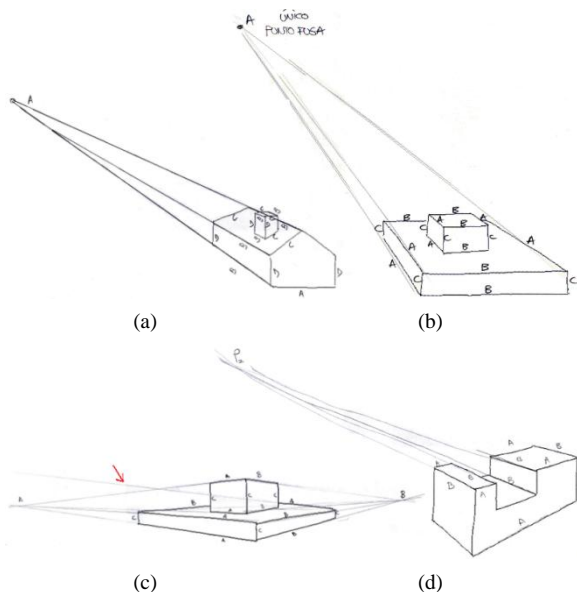


Figure 10. Strategies to select the most likely VP location

Firstly, some subjects selected the most likely lines in the group and use them to find the VP. Subjects in this group used different strategies to select the most representative lines: (a) select the outer lines (those which encompass the whole group, as in Figure 10a), (b) to discard the shortest lines (Figure 10b) and the most erratic lines (Figure 10c); or (c) simply estimate a rough VP location using a random subset of the lines (Figure 10d). This strategy was most commonly used for sketches 2 (Fig. 10b), 3, 5 (Fig. 10d), 7, 10 (Fig. 10a), 13 (Fig. 10c) and 14.

Secondly, some subjects calculated more than one location for the same VP. These subjects identified the convergence in two ways: (a) by means of a point cloud formed by intersections of the lines of the same group (Figure 11a), or 2) defining different groups of lines which belong to the same axis (Figure 11b). This strategy was most commonly used for sketches 9 (Figure 11b), 12 (Figure 11c), 13 (Figure 11d) and 18 (Figure 11e).

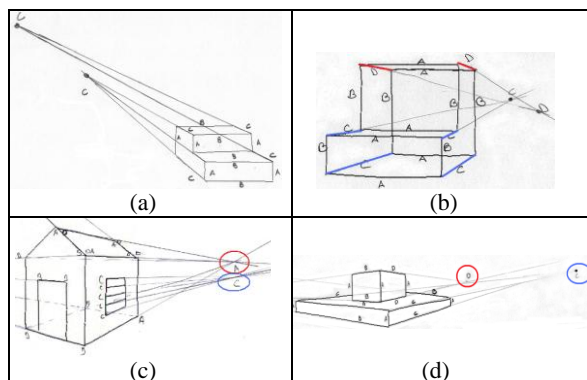


Figure 11. Point clouds for VPs definition

Thirdly, some subjects seemed to detect convergence but did not locate any VPs.

Thus, our second hypothesis should be rejected: humans do not seem to be able to locate VPs precisely: they agree about the orientation angle of the VP, but not about its position along this line.

Qualitative validation of the third hypothesis

We note that the uncertain cases in Table I (those in the range 50-75%) are typically those formed from either a) groups of lines which contain the shortest lines of the drawing, or b) groups with low density of lines. This perception supports our third hypothesis: that length of lines influences the convergence detection. However, in the light of the data, we must also take into account the density of the group of lines.

6. Numerical measurement

Having evaluated our hypotheses qualitatively, we now search for metrics which can help to tune automatic algorithms for finding VPs in engineering

sketches. We define and study some geometric parameters.

Firstly, since our goal is finding parameters which influence in the perception success, we shall compare our geometric parameters with the parameter used in Table I:

Perception degree (Det) is the percentage of the subjects who identified a VP for a specific sketch and axis.

$$\text{Det} = (\text{No. detections} / \text{No. Questionnaires}) * 100 \quad (1)$$

Some geometric parameters must be normalised to avoid the influence of the drawing size. We use the radius of the sketch's bounding circle. We calculate this as the maximum distance between the centre of mass of the sketch and any of its vertices. First, the centre of mass of the sketch is calculated (x_{com}, y_{com}) .

$$x_{com} = (\sum x_i) / n_v, \quad y_{com} = (\sum y_i) / n_v \quad \forall i \in \mathbf{V}_S \quad (2)$$

where n_v represents the number of vertices of the sketch S and \mathbf{V}_S is the set of vertices of sketch S .

$$\text{size} = \max((x_i - x_{com})^2 + (y_i - y_{com})^2)^{1/2} \quad \forall i \in \mathbf{V}_S \quad (3)$$

In order to evaluate metrics related to the first hypothesis we analysed the convergence by means of the following parameter:

Angular dispersion (AD) is the maximum aperture angle between pairs of lines in a group of lines which represent parallel edges in space:

$$\text{AD} = \max(|\alpha_i - \alpha_j|) \quad \forall i, j \in \mathbf{L}_{SA} \quad (4)$$

where α_i and α_j represent, respectively, the angles of edges i and j relative to the same origin; and \mathbf{L}_{SA} is the group of lines of sketch S and axis A .

In order to evaluate metrics related to the second hypothesis we define the following parameters:

Dispersion (Disp) measures the density of the point cloud of VPs located by the subjects (such as the blue, green and red clouds in Figure 8).

First, the centroid of the cloud is calculated:

$$x_{centroid} = (\sum x_i) / n_{vp}, \quad y_{centroid} = (\sum y_i) / n_{vp} \quad \forall i \in \mathbf{VP}_{SA} \quad (5)$$

where \mathbf{VP}_{SA} is the cloud of VPs for sketch S and axis A , and n_{vp} is the size of this cloud. Next we calculate the Euclidean distances d_i between each point (x_i, y_i) in the cloud and its centroid $(x_{centroid}, y_{centroid})$.

$$d_i = [(x_i - x_{centroid})^2 + (y_i - y_{centroid})^2]^{1/2} \quad \forall i \in \mathbf{VP}_{SA} \quad (6)$$

Then, the standard deviation of these distances (dev) is calculated.

$$Av = (\sum d_i) / n \quad \forall i \in \mathbf{VP}_{SA} \quad (7)$$

$$dev = [(1/(n-1)) * \sum (d_i - Av)^2]^{1/2} \quad \forall i \in \mathbf{VP}_{SA} \quad (8)$$

Finally, to avoid the influence of the drawing size, the parameter is normalised.

$$\text{Disp} = (dev / \text{size}) * 100 \quad (9)$$

Distance ratio between centroids (DRC) measures how far away the point cloud is from the sketch. It is calculated as the Euclidian distance between the centre of mass of the sketch and the centroid of the point cloud:

$$\text{Dist} = ((x_{centroid} - x_{com})^2 + (y_{centroid} - y_{com})^2)^{1/2} \quad (10)$$

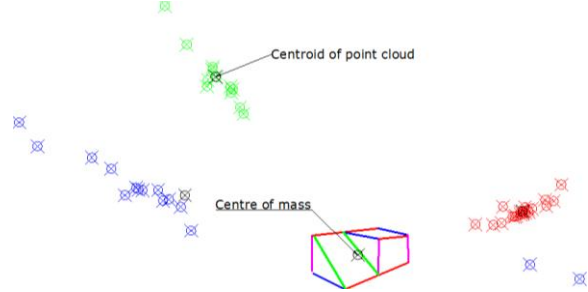


Figure 12. DRC calculation for Sketch 3, Axis 4

Finally, to avoid the influence of the drawing size, the parameter is normalised:

$$\text{DRC} = \text{Dist} / \text{size} \quad (11)$$

In order to evaluate metrics related to the third hypothesis we define the following parameters:

Length Dispersion (LeD) is a standard deviation which measures the dispersion of the lengths of the lines (l_i) which belong to the same sketch and axis: the more the lengths of lines differ, the higher its value. First the average length L_{aver} is calculated. Taking l_i as the length of each line and n_l as the number of lines:

$$L_{aver} = (\sum l_i) / n_l \quad \forall i \in \mathbf{L}_{SA} \quad (12)$$

$$\text{LeD} = [(1/(n_{SA}-1)) * \sum (l_i - L_{aver})^2]^{1/2} \quad \forall i \in \mathbf{L}_{SA} \quad (13)$$

where n_{SA} is the number of lines in the group of parallel edges of sketch S , axis A .

Number of lines (NL) which belong to the same sketch and axis.

$$\text{NL} = n_{SD} \quad (14)$$

Location Dispersion (LoD) is a standard deviation which measures the influence of dispersion of the locations of the midpoints of lines which belong to the same sketch and axis) and the normalised length of each line. The more the locations of lines differ, and the shorter the lines, the higher the value.

We first compute the location of each midpoint, by way of the head (x_h, y_h) and tail (x_t, y_t) coordinates of each line:

$$\text{loc}_i = ((x_h + x_t) / 2, (y_h + y_t) / 2) \quad \forall i \in \mathbf{L}_{SA} \quad (15)$$

$$\text{Loc}_{aver} = (\sum \text{loc}_i) / n_l \quad \forall i \in \mathbf{L}_{SA} \quad (16)$$

Next we calculate the standard deviation normalised with the relative length:

$$\text{LoD} = [(1/(n_{SD}-1)) * (\text{size}/l_i) * \sum (\text{loc}_i - \text{Loc}_{aver})^2]^{1/2} \quad \forall i \in \mathbf{L}_{SA} \quad (17)$$

We calculated all of these parameters for each sketch and axis. The results are shown in the Table II, which is arranged in decreasing order of values of convergence detection.

Sketch	Axis	Det	AD	Disp	DRC	LeD	NL	LoD
3	1	100.00	16.99	27.54	3.15	12.61	3	7.23
4	1	100.00	27.95	51.92	2.19	1.91	6	4.76
5	1	100.00	15.25	45.26	3.46	14.46	7	37.17
6	1	100.00	31.67	35.54	2.03	3.34	6	6.85
9	1	100.00	51.51	50.89	1.69	7.44	6	50.18
12	1	100.00	41.24	22.31	1.91	8.98	8	10.69
13	1	100.00	23.73	32.11	1.72	12.19	5	20.11
16	2	100.00	17.42	23.30	3.18	1.67	5	7.04
17	1	94.74	16.12	44.48	3.66	9.33	4	9.99
8	2	94.44	15.23	46.23	4.70	1.58	5	5.06
2	2	94.12	24.65	26.51	2.16	12.09	6	22.61
7	2	93.75	16.85	39.22	3.70	11.47	3	7.66
1	2	93.33	16.35	39.05	3.46	4.39	5	14.87
10	2	92.86	16.32	96.48	4.00	13.19	7	26.21
13	2	92.31	14.21	36.53	2.11	20.59	5	25.36
6	2	85.71	15.12	30.89	2.93	8.95	6	22.06
17	3	84.21	21.27	200.23	3.53	2.43	4	12.89
5	2	84.21	26.42	78.50	2.82	0.95	6	14.84
14	1	83.33	13.57	40.21	3.02	8.37	6	16.51
6	3	78.57	21.04	86.08	4.18	6.30	5	29.35
11	2	77.78	7.24	25.93	3.11	5.15	2	4.82
4	2	76.92	27.29	194.10	1.64	5.11	6	27.50
12	2	76.47	12.20	33.43	3.14	11.17	2	13.92
7	4	75.00	17.07	62.57	3.92	4.18	4	5.38
14	3	75.00	44.09	20.41	1.75	0.94	6	90.77
3	2	72.22	15.31	224.17	3.39	1.67	3	14.74
8	1	72.22	14.05	161.48	5.94	4.84	5	14.88
10	4	71.43	16.45	26.16	2.38	6.73	3	7.32
7	1	68.75	33.87	12.41	2.33	2.73	4	5.26
18	3	68.75	19.70	63.05	3.49	4.63	8	61.74
17	4	68.42	5.95	68.31	4.82	3.57	2	0.67
3	4	66.67	8.52	44.61	4.22	2.27	2	0.60
15	2	57.14	6.23	86.23	4.19	6.14	3	14.47
1	1	46.67	1.55	545.13	8.02	2.67	5	6.09
11	4	44.44	8.93	20.18	5.63	2.44	2	5.33
15	5	42.86	3.90	164.41	7.14	1.41	2	4.62
12	4	35.29	6.10	15.12	3.31	4.26	2	7.94
1	3	33.33	6.36	197.15	4.61	2.12	5	18.98
11	5	33.33	1.62	113.02	7.52	2.60	2	9.45
17	2	31.58	4.60	479.19	1.99	2.61	4	14.12
15	1	28.57	1.06	0.00	18.07	1.47	5	2.73
15	3	28.57	2.12	1429.85	20.07	2.38	6	21.68
15	4	28.57	2.29	0.00	16.70	0.61	2	1.91
2	3	23.53	7.03	144.20	2.52	1.46	6	92.01
11	1	22.22	1.92	0.00	1.79	4.09	5	2.43
2	1	17.65	3.82	745.68	12.49	21.37	6	83.56
18	2	12.50	4.18	0.00	1.78	24.38	8	9.63
14	2	8.33	1.54		3.38	19.12	6	19.98
9	2	7.14	1.17		4.15	5.73	6	8.66
10	1	7.14	5.29		2.31	13.76	3	30.75
16	1	6.67	3.73		5.76	7.29	5	23.03
7	3	6.25	3.26		12.89	4.67	4	10.96
18	1	6.25	5.89		2.41	29.39	8	16.92
12	3	5.88	1.61		2.18	9.51	7	17.04
12	5	5.88	0.00		3.26		1	
3	3	5.56	1.71		10.14	2.60	3	19.86
5	3	5.26	4.08		4.71	6.71	6	17.25
4	3	0.00	1.88			8.98	6	43.72
8	3	0.00	1.91			4.49	5	17.59
9	3	0.00	1.76			8.11	6	26.04
10	3	0.00	2.82			2.52	6	30.36
10	5	0.00	0.00				1	
11	3	0.00	1.76			1.53	4	35.24
13	3	0.00	3.27			4.88	6	252.70
16	3	0	2.05			8.16	5	28.28

Table II. Parameters which influence perception success

7. Analysis

A statistic analysis based on Pearson correlation shows the mutual influence of each pair of parameters. For this study we omitted the cases where the group was a single line (NL= 1), as they give no useful information (specifically, we omitted sketch 10 axis 5 and sketch 12 axis 5, although, as it was showed in Fig. 9, some people included hidden lines for sketch 12 axis 5 to get the information they needed to locate the VP).

	Det	AD	Disp	DRC	LeD	NL	LoD
Det	Pear 1	.757**	-.378*	-.403**	-.054	-.044	-.222
	Sig.	.000	.009	.002	.672	.733	.080
	N	63	63	47	56	63	63
AD	Pear		1	-.295*	-.460**	-.064	.229
	Sig.			.044	.000	.621	.071
	N			63	47	56	63
Disp	Pear			1	.581**	-.028	.117
	Sig.				.000	.854	.433
	N				47	47	47
DRC	Pear				1	-.219	-.181
	Sig.					.104	.181
	N					56	56
LeD	Pear					1	.379**
	Sig.						.002
	N						63
NL	Pear						1
	Sig.						
	N						
LoD							
							1
							63

Table III. Pearson correlation among parameters

We analyse our hypotheses in the light of these results.

Hypothesis 1

Table III shows that detection degree (Det) correlates best with AD (as was qualitatively deduced in Section 5.2). From Table II, we can see that, for directions where Det > 75%, the minimum AD value is 7.24° (sketch 11 axis 2, or figure 13a). This could be used as a minimum threshold, particularly since it is close to the maximum threshold of 8° proposed in [Plu10] for considering a bundle of lines as parallel. However, our qualitative analyses found sketch 11 to be uncertain. If we exclude this result, we get a more conservative minimum value of AD = 12.2° (sketch 12 axis 2, or figure 13b).

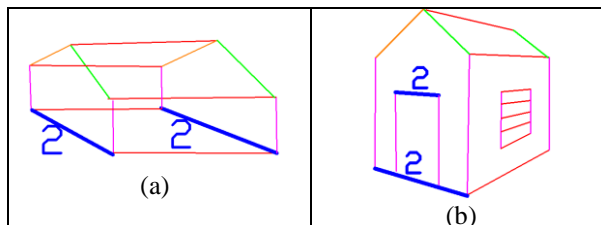


Figure 13. Sketches 11 and 12

We conclude that people clearly and consistently perceive convergence in spite of sketch imperfection

if the lines span an angle of at least 12° . Between 12° and 8° the perception is uncertain. Hence, an algorithm for finding VPs should give a high probability to bundles spanning at least 12° ; the probability should decrease between 12° and 8° , and should be close to zero below 8° .

Considering the uncertain cases of table II (those detected between 50% and 75%), three of them (sketch 17 axis 4, sketch 3 axis 4 and sketch 15 axis 2) have AD values close to 8° , as predicted by the above criterion. However, other cases (sketch 3 axis 2 (Fig. 14a), sketch 8 axis 1 (Fig. 14b), sketch 10 axis 4 (Fig. 14c), sketch 7 axis 1 (Fig. 14d) and sketch 18 axis 3 (Fig. 14e) have AD higher than 12° , which, according to our criterion, should encourage subjects to perceive them as unambiguously convergent. We note that all of these contain the shortest lines of their sketches (sometimes alongside medium-length lines), and in addition they have a low line density (i.e. not only is the number of lines low—3 or 4 lines, except sketch 8, which has 5 lines for axis 1—but they are dispersed around the sketch rather than clustered together). This seems to support our third hypothesis, that the lengths of lines will influence perception of their convergence.

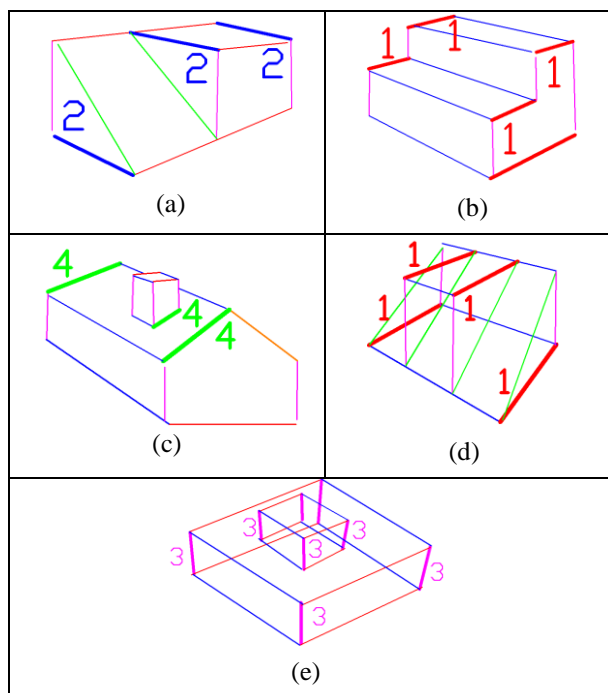


Figure 14. Sketches and axis

Hypothesis 2

As argued in section 5.2, humans are able to agree about the orientation angle of a VP but not about the precise location of the VP along this line. This pattern can be quantified by means of DRC and its dispersion measure Disp.

Table III shows that detection degree (Det) correlates with small values of distance ratio between centroids (DRC), and usually with small dispersion between the intersection points (Disp).

From Table II, we see that, for those VPs perceived by all subjects (Det=100), the value of DRC varies between 1.6 and 3, except in those cases where the group of lines includes the largest lines of the drawings (sketch 3 axis 1, sketch 5 axis 1, sketch 16 axis 2), in which cases the length of the lines seems to encourage subjects to locate the VP further away, and the value of DRC rises to 3.46. These locations correspond to small or medium values of the dispersion point cloud (with Disp between 22.31% and 51.92%), which means that people agree to locate the VPs within a small area.

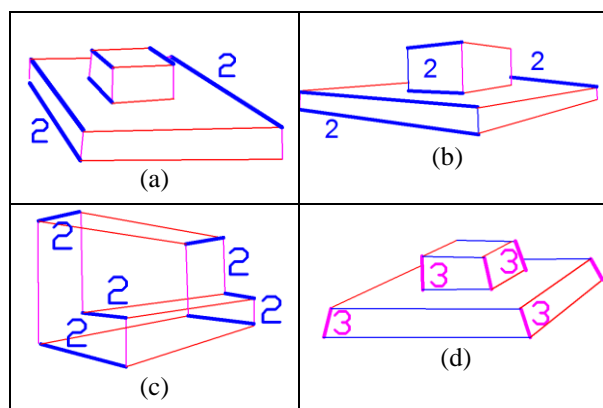


Figure 15. Sketches and axis

For Det in the range 99–75, the value of DRC is generally in the range 3 to 5. The exceptions are those examples which are uncertain or have a DRC value slightly lower than 3. In both Sketch 2 axis 2 (Fig. 15a) and sketch 13 axis 2 (Fig. 15b), the difference of line lengths is evident, and the shortest lines form a visual group distinct from the longest lines. The other exceptions are sketch 4 axis 2 (Fig. 15c), sketch 14 axis 3 (Fig. 15d), sketch 6 axis 2 and sketch 5 axis 2 in which short or medium lines are dispersed around the drawing.

Cases with detection under 50% generally have values of DRC higher than 5, except sketch 12 axis 4 (two lines in an oblique direction, Fig. 16a) and sketch 1 axis 3 (Fig. 16b), sketch 17 axis 2 (Fig. 16c) and sketch 2 axis 3 (Fig. 16d), which contain short and medium lines, far apart in the drawing.

Cases with detection under 50% also generally have values of “Disp” over 100%, except for two cases of oblique axes (sketch 11 axis 4 and sketch 12 axis 4, Fig. 16a) where subjects seem to reach agreement about the locations.

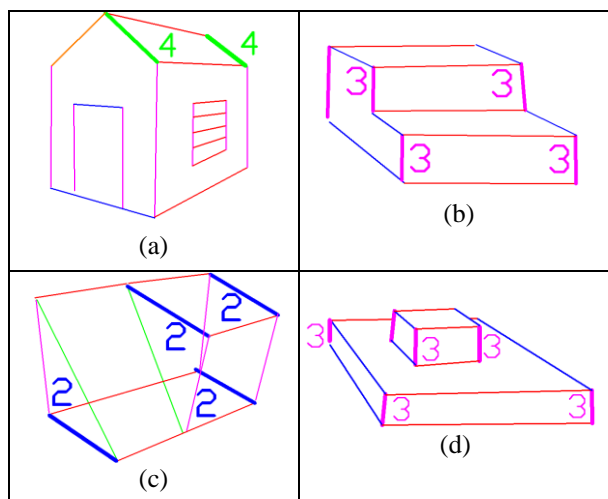


Figure 16. Sketches and axis

We do not take into account those cases with very low point cloud density, since the information is insufficient to extract any significant conclusion. For example, the closeness of the centroid of the point cloud to the sketch for sketch 11 axis 1 (Fig. 17) is clearly irrelevant.



Figure 17. Point cloud for sketch 11 axis 1.

In summary, we found that DRC between 1.6 and 3 a VP is likely to be perceived. With DRC between 3 and 5 a VP is somewhat less likely to be perceived. With DRC higher than 5, lines are not perceived as convergent.

Hence, an algorithm for finding VPs should assign a high probability to candidate VPs located in a ring whose minimum radius is 1.6 times the radius of the bounding circle, and whose maximum radius is 3 times the radius of the bounding circle. The probability should decrease outside this ring, and be close to zero outside an outer ring whose radius is 5 times the radius of the bounding circle.

In cases where the lines are short and visually apart from one another, people’s behaviour changes. This is considered under the third hypothesis.

Hypothesis 3

To evaluate our third hypothesis, we focus on the standard deviation between lengths LeD. Table III shows no relationship between the detection of convergence (Det) or the answers dispersion (Disp) and LeD. Should we then simply reject hypothesis 3? We cannot deny the influence of differences in line length, since we have already noticed that subjects changed their answer patterns under specific situations:

- a) When the group of lines contains lines of very different lengths.
- b) When the group of lines includes only short lines and they are dispersed through drawing.
- c) When number of lines in the group is low.

Thus we should consider LoD as a secondary parameter, whose influence appears only in certain cases.

According to table II, for values of LoD higher than 70 (short lines widely dispersed around the sketch), our subjects generally followed the criteria described in previous hypotheses, and parameters AD and DRC reflect the response of human perception.

LoD values below 5 may result from long lines with a homogeneous location in the drawing (as in Fig. 18). However, a low value may also be due to low number of lines, which distorts the standard deviation measure. For groups of fewer than 4 lines, this parameter may not give reliable information.

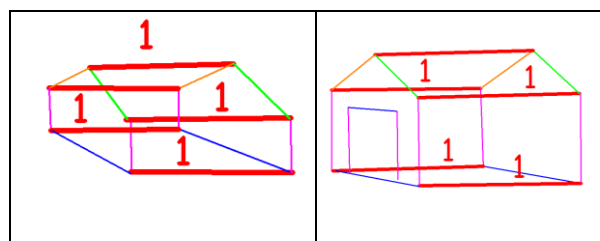


Figure 18. Sketches 11 and 15 with low value of LoD.

LoD in the range 5-70 seems to influence human perception in two different ways: (i) sometimes the detection degree is slightly different from that predicted by parameters AD and DRC—for example, in sketch 18 axis 3 (Fig. 14e), it seems that in spite of having an AD higher than 12, the separation between the lines, which are also the shortest in the sketch, prevents humans from perceiving the convergence (25% of subjects said it was an axonometric drawing); (ii) at other times the location agreement decreases which results in higher values of Disp.

Sketch	Axis	Det	AD	Disp	DRC	LeD	NL	LoD
15	3	28.57	2.12	1429.85	20.07	2.38	6	21.68
2	1	17.65	3.82	745.68	12.49	21.37	6	83.56
1	1	46.67	1.55	545.13	8.02	2.67	5	6.09
17	2	31.58	4.60	479.19	1.99	2.61	4	14.12
3	2	72.22	15.31	224.17	3.39	1.67	3	14.74
17	3	84.21	21.27	200.23	3.53	2.43	4	12.89
1	3	33.33	6.36	197.15	4.61	2.12	5	18.98
4	2	76.92	27.29	194.10	1.64	5.11	6	27.50
15	5	42.86	3.90	164.41	7.14	1.41	2	4.62
8	1	72.22	14.05	161.48	5.94	4.84	5	14.88
2	3	23.53	7.03	144.20	2.52	1.46	6	92.01
11	5	33.33	1.62	113.02	7.52	2.60	2	9.45
10	2	92.86	16.32	96.48	4.00	13.19	7	26.21

Table IV. Perceived vanishing directions

Table IV is a sub-table of table II rearranged in descending order of Disp. From this table, it appears

that very high values of Disp (Disp>500) seem to correspond to nearly parallel groups of edges (AD<4). Values of Disp in the range (100 to 500) seem to appear as a combined effect of small number of lines (NL<=3), and/or high dispersion of the location of lines LoD \in (5, 70).

Thus, dispersion is high for uncertain cases which have small values of AD, or when the set of lines contains short and medium lines dispersed through the sketch. However, we have previously noted that when the group of lines contains lines of very different lengths people usually apply the strategies illustrated in figure 10 (using outermost lines, ignoring short or erratic lines) - the most common is ignoring the shorter lines, and using only the longer lines to locate the vanishing point - and in such cases it seems that the dispersion decreases.

This strategy is easy to replicate algorithmically and it would model human perception well. The most representative cases (sketches 2, 5, 10, 13, illustrated in Fig. 10, and sketches 3, 7 and 14) have values of Led higher than 10. Thus Led > 10 could be used as a threshold for signalling to the algorithm that it should use only long lines to locate the VP.

8. Conclusions

Current image analysis approaches take 2D camera images as their input, so do not solve satisfactorily the problem of geometrical imperfections inherent in sketches. At this end we have conducted a pilot experiment which gives us preliminary criteria and metrics for implementing algorithms which mimic human perception in detecting vanishing points in design sketches.

Human beings can perceive the existence of intended vanishing points in sketches of 3D polyhedral shapes, in spite of their inherent imperfections. Humans generally perceive vanishing points for sets of lines spanning 12 or more degrees.

Humans agree about the orientation angle of the VP relative to the sketch. They often do not agree about the distance of the VP from the sketch. VPs are easiest to perceive and to locate if they are neither too close nor too far away from the sketch: ideally, at distances not much more than the size of the sketch. (We can hypothesise that sketches are produced according to these expectations.) Algorithms should follow these perceptual criteria: enforcement of the acceptance criteria should be tolerant to imperfections inside the main region (1.6x to 3x), and stricter outside (3x to 5x).

A number of secondary parameters combine to influence the perception and location of VPs, including line length, lines location and density of lines. The influence of these “distractions” can be subtle, and remain a matter for future research.

9. Acknowledgments

Support from the Spanish Ministry of Science and Education and the European Union (Project DPI2007-66755-C02-01) and the Ramon y Cajal Scholarship Programme are acknowledged with gratitude.

10. REFERENCES

- [Com04] Company, P., Contero, M., Conesa, J. and Piquer, A. An optimisation-based reconstruction engine for 3D modelling by sketching. *Computer and Graphics* 28, 955-979, 2004.
- [Gol99] Goldstein, E.B. *Sensation & Perception*. Wadsworth Publishing Company, 1999, ISBN 0-534-34680-4
- [Hof00] Hoffmann; D. *Visual Intelligence. How we create what we see*, New York: WW Norton & Company, 2000.
- [Joh09] Johnson, G., Gross, M.D., Hong, J. and Yi-Luen Do, E., *Computational Support for Sketching in Design: A Review, Foundations and Trends in Human-Computer Interaction*, 2(1), 1-93, 2009.
- [Lip96] Lipson, H. and Shpitalni, M. Optimization-based reconstruction of a 3D object from a single freehand line drawing. *Computer-Aided Design*; 28(8):651-63, 1996.
- [Pal99] Palmer, SE., *Vision science, Photons to phenomenology*, Cambridge, MA: The MIT Press, 1999.
- [Plu10] Plumed, R., Company, P., Piquer, A. and Varley, P.A.C. Do engineers use convergence to a vanishing point when sketching? *Proc. Int. Symposium on Distributed Computing and Artificial Intelligence 2010, (DCAI'10)*, pp.241-250, 2010.
- [Tia09] Tian, C., Masry, M. and, Lipson, H. Physical sketching: Reconstruction and análisis of 3D objects from freehand sketches. *Computer aided design*; 41: 147-158, 2009.
- [Var03] Varley, P.A.C. *Automatic Creation of Boundary-Representation Models from Single Line Drawings*, PhD Thesis, University of Wales, 2003.
- [Wri83] Wright, L., *Perspective in Perspective*, Routledge & Kegan Paul, London, 1983.
- [Yua08] Yuan S., Tsui L.Y. and Jie S. Regularity selection for effective 3D object reconstruction from a single line drawing. *Pattern Recognition Letters* 29 (10), 1486-1495, 2008.

Efficient complementary viewpoint selection in volume rendering

Sergi Grau
Dept. Llenguatges i
Sistemes Informatics,
Polytechnic University of
Catalonia.
sgrau@lsi.upc.edu

Anna Puig, Sergio
Escalera, Maria Salamó
Dept. Matemàtica
Aplicada i Anàlisi,
Universitat de Barcelona.
{anna,sergio,maria}@maia.ub.es

Oscar Amorós
Dept. Matemàtica
Aplicada i Anàlisi,
Universitat de Barcelona.
morousg@gmail.com

ABSTRACT

A major goal of visualization is to appropriately express knowledge of scientific data. Generally, gathering visual information contained in the volume data often requires a lot of expertise from the final user to setup the parameters of the visualization. One way of alleviating this problem is to provide the position of inner structures with different viewpoint locations to enhance the perception and construction of the mental image. To this end, traditional illustrations use two or three different views of the regions of interest. Similarly, with the aim of assisting the users to easily place a good viewpoint location, this paper proposes an automatic and interactive method that locates different complementary viewpoints from a reference camera in volume datasets. Specifically, the proposed method combines the quantity of information each camera provides for each structure and the shape similarity of the projections of the remaining viewpoints based on Dynamic Time Warping. The selected complementary viewpoints allow a better understanding of the focused structure in several applications. Thus, the user interactively receives feedback based on several viewpoints that helps him to understand the visual information. A live-user evaluation on different data sets show a good convergence to useful complementary viewpoints.

Keywords

Dual camera; Visualization; Interactive Interfaces; Dynamic Time Warping.

1 INTRODUCTION

One of the most important visualization goals is to appropriately express knowledge of scientific data. During the last few decades several methods have been published in the bibliography to gather visual information contained in the data. However, gathering visual information often requires the expertise from the final user in a difficult and tedious process in order to setup the parameters of the visualization. Some metaphors of interaction have been provided to help users in data navigation between focus and context (importance-driven, VolumeShop, exoVis, LiveSync++, ClearView).

Over the centuries, the traditional paradigm used in illustration for visual abstraction has been to enhance the most important structures into an environment or context with different painting techniques. As an example, Fig. 1 shows two traditional illustrations which use two or three different views of the region of interest to enhance the construction of the mental image. Particularly, in Fig. 1(a) the focused structure is an eye which shows two complementary viewpoints, an oblique view and a frontal view, and Fig. 1(b) illustrates complementary views of the foot.

Previously, we proposed an enhancement of the ClearView paradigm where the context's volume data

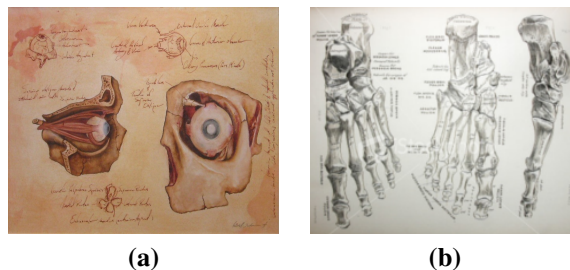


Figure 1: Illustration examples of the complementary camera viewpoint. Images (a) and (b) come from <http://www.keithtuckerart.com/Illustration.html> and the medical dictionary Allen's Anatomy, respectively.

is adaptively clipped around the focus region [GP09]. Apart from the focus, to select a good starting viewpoint of the focus is a difficult task as volume renderings often include a barrage of complex 3D structures that can overwhelm the user. Several methods have been published to gather visual information contained in the data. There are methods in the field of illustrative visualization [RBGV08], such as [VKG05] and [BG05], where the main goal is to develop applications that can integrate illustrations in the expert's ordinary data analysis in order to get more semantics from the data.

Once a good focus and context is obtained, it is also important to concentrate on alleviating the normally difficult abstraction process that a user should carry out to convey the desired information in the underlying data. Specifically, to select a good starting viewpoint is sometimes a tedious task for a non-experimented users. Actually, in the clinical routine prefixed views are used based on sagittal, axial and coronary views, despite that they are not always the best. In the bibliography, we can find techniques that automatically locate the viewpoint according to the importance of the structure to be rendered [BS05a, TFTN05, PPVN08]. Most of them are computed in a preprocessed stage due to their high computational cost.

In this paper we propose the computation of the correlated view that shows highly complementary information of the focused structures. Also, in our proposal the user can adjust suggested cameras interactively, and our system computes the new complementary view. To this end, we consider both the visual information and the shape similarity of the projections of each sampled camera. Similarity is based on Dynamic Time Warping, ranking the projections using a weighted similarity related to the reference camera. In this manner, the proposal is able to cope with the problem of locating automatically the complementary camera that maximizes the complementary information of the structures. By its nature, the proposed algorithm is general enough to be applied in combination to state-of-the-art approaches for selecting the optimal camera. Based on a live-user evaluation, we show how the proposal efficiently computes complementary viewpoints and rapidly converges to a good solution.

The rest of the paper is structured as follows: Section 2 describes relevant previous work on viewpoint selection. Our proposal is described in Section 3. Section 4 analyzes the proposal with different data sets. Finally, Section 5 concludes the paper.

2 RELATED WORK

The cognitive process to comprehend the meaningful structures in an image varies according to the visual purpose. A single visual event is easily processed in an intuitive way. However, multiple visual events involve a more complex cognitive process. Several studies have addressed this problem. This section summarizes previous works on visual information measures used to define the best viewpoint selection.

Setting the camera to focus on the relevant structures of the model sometimes requires a lot of user expertise. Therefore, many authors [BS05a, TFTN05, VFSG06, MNTP07a, PPVN08, KBKG08, ZWD12] have addressed viewpoint optimization in direct volume rendering, extending the ideas used in surface-based scenes.

Visual information is the measure used to estimate the quality of a viewpoint according to the perception of the structures of interest of a volume data set. Then, information theory approaches, such as viewpoint entropy [BS05a], generalize the analysis of the visual information associated to a specific viewpoint. Particular cases are the approaches focused on heuristic functions [KBKG08, TFTN05, MNTP07b, CQWZ06, GP09]. These metrics are not universal and, as [PPB*05] concludes, not one descriptor performs a perfect job. Several descriptors to measure the visual information associated to a view projection have been previously described in the literature. Heuristic functions and entropy-based methods mainly consider three types of descriptors: environment-dependent [KBKG08, PPVN08], object-dependent [VFSG06, BS05a, KBKG08] and viewpoint-dependent [ZWD12].

First, environment-dependent descriptors depend on contextual parameters such as patient orientation and viewpoint history [KBKG08, PPVN08]. Object-dependent descriptors characterize the information contained in the data set, such as the importance of the selected objects [VFSG06], the noteworthiness of each voxel that contributes in the final image [BS05a], the shape of the selected object [KBKG08], and the use of heuristics for complementary views [GP09]. These type of descriptors depend upon two main requirements: a complete model of each view and the position of the camera. The process of all these models involves a high computational cost that is mostly alleviated by introducing a preprocessing stage for each view. Finally, viewpoint-dependent descriptors are based on the visibility and the location of the final projection of the object in the viewport. They measure only the information that will be effectively seen by the user. The feature most considered are the size of the projected area of the object [MNTP07b], the occlusion between objects [MNTP07b, KBKG08], object self-occlusions [CQWZ06], the important viewport areas [VFSG06] and the entropy of the image [PPVN08]. Some approaches study correlations between cameras in varying time measuring the stability between the views based on the Jense-Shanon divergence metric [BS05b]. Also, in path-views searching a Normalized Compression Distance is used [PPVN08]. They search the next best view at the greatest distance from the previous one. [CMH08] proposes a global search of optimal points in the solution space using potential fields.

Our main goal in this work is to measure the information that will be effectively seen by the user in two correlated views. The requirements are that the measure should be simple, easy to evaluate, robust to changes in the resolution of the final view, and without user intervention. Viewpoint-dependent descriptors are the

most suitable since they are focused on the final visual user's perception. They are easy to compute and they do not require a preprocess stage such as object-dependent ones. In addition, we propose to measure a new descriptor based on the shape similarity of the projected structures that complements the projected area information. We present a 2D-geometrical approach based on Dynamic Time Warping that matches distortions between the projected shapes. This method is based on implicit distance functions that are stable and robust to shape perturbations and noise. Thus, our proposal obtains a representative complementary projection using a viewpoint-dependent approach, i.e. analyzing the visual information associated to each view by an entropy-based method that combines the entropies of each projected visible structure and shape similarities.

3 LOCATING COMPLEMENTARY VIEWPOINTS

The main goal of our system is to provide an interactive exploration of volume data sets by unexperienced users. To visualise the volume, we use an interactive interface, allowing users to both understand and control aspects of the visualization process that would otherwise go unnoticed.

In order to define the location of the complementary camera of the underlying structures of the data from a reference viewpoint, v_r , we calculate the entropy of a set of sampled cameras, $H(v)$. Next, we analyze the shape similarity, Sim , of the viewpoint projections related to the v_r viewpoint based on Dynamic Time Warping. The weighted combination of these measures produces the final complementary view, $Dual(v_r)$.

3.1 Visual information descriptor

We use an entropy-based method to estimate the quality of a camera based on the projected area of the isosurfaces of the selected structures. Multiple focused regions are considered in the visual information estimation. We calculate the weighted sum of the viewpoint entropy's of the extracted structures. We assign higher weights to these structures with opacity transfer functions, as in [TFTN05]. Let $F = \{f_1, f_2, \dots, f_j\}$ be the set of user-selected structures, or features, we define the function, $w(f_j)$, as the weight associated to the each feature, f_j . Let $H_{f_j}(v)$ be the entropy of the viewpoint, v , associated to the f_j feature. Thus, the final entropy associated to a viewpoint, v , is defined as:

$$H(v) = \sum_{j=1}^{|F|} H_{f_j}(v) \cdot w(f_j). \quad (1)$$

The entropy $H_{f_j}(v)$ is based on view-dependent descriptors. In the bibliography, the most used are the

size of the projected area of the object [PPVN08], the occlusion between objects [KBKG08], object self-occlusions, the important viewport areas [VFSG06], and the luminance [ZWD12]. In this proposal, without loss of generality, we use as a proof of concept, the projected area of the voxels of the feature f_j .

Finding a good view assumes that the data set is centered at the origin and the camera is restricted to be at a fixed distance from the origin, defining a boundary sphere. The surface of this sphere represents all view directions. Thus, let S be the 3D space which represents the bounding sphere of the complete model that contains the set of camera locations or viewpoints. Let $V = \{v_1, v_2, \dots, v_N\}$ be the space of $|V|$ camera locations defined as a discrete and finite set of iso-distributed samples over the surface of S . To guarantee a regular distribution we use the HealPix package [GHB*05]. An example of a sphere and the iso-distributed samples over the surface is shown in Fig. 2, highlighting the images related to each viewpoint.

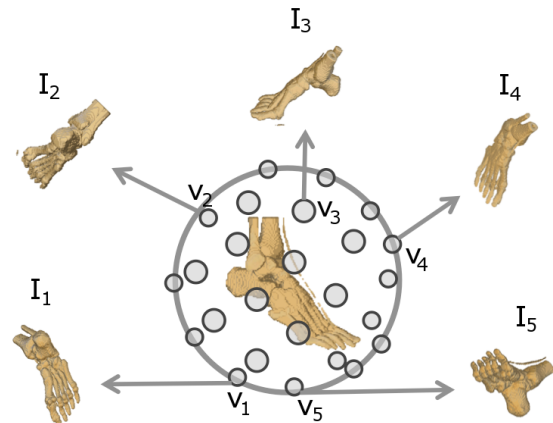


Figure 2: Description of the sampled sphere, S , each image I_i is related to each viewpoint v_i .

We assume that the up vector of the cameras can be arbitrarily chosen at each location due to the camera rolls not having an effect on the visibility. Each view projection of the selected features is a 2D image, I_i . Let $I = \{I_1, I_2, \dots, I_N\}$ be the set of all images projected over the sphere and $H(v_i)$ their corresponding entropy. Generally, focused regions can be completely occluded by others, and this fact can difficult the computation of the final entropy of a viewpoint. We can assume that each I_i is the set of layers where each one is the projection of all the voxels of the f_j feature without occluders. These layers could be used to compute the $H_{f_j}(v)$ entropy, and also to define the shape similarity function.

3.2 Shape similarity term

Once we have the camera locations and a reference 2D image I_r projected from the reference camera viewpoint, the objective is to find a complementary image

$I_i \in I$ so that it offers the most complementary information to the reference image I_r . For this task, an image similarity procedure among I_r and each candidate I_i is computed iteratively. In order to perform this estimation, we first need a definition about what we refer for similarity. Although different features and definitions can be considered for this task, we just will use simple image information that can be useful to discriminate similar visual shapes in terms of silhouette and region properties. In particular, we focus on the external contour similarity of 3D object projections as well as the amount of visual information of each volume structure from a particular point of view. As we will see, it allows for a simple estimation that is effective on selecting an useful complementary camera.

The similarity measure used to compute this information is obtained as follows:

$$Sim(I_r, I_i) = (1 - \beta) \cdot Dist(I_r, I_i) + \beta \cdot Corr(I_r, I_i), \quad (2)$$

where Sim stands for the similarity measure between a pair of images, $Dist$ (see Eq. 3) is the function that computes a shape similarity among a pair of images, and $Corr$ (see Eq. 6) is the function that computes the correlation among percentage of visible information for each structure present in both images. Finally, β is a regularization factor. $Sim(I_r, I_i)$ tends to 0 when I_r and I_i are close to each other. On the other hand $Sim(I_r, I_i)$ tends to 1. Note that for simplicity, we consider shape contour and label properties information in order to compute shape similarity.

For the shape similarity term $Dist(I_r, I_i)$, our goal is to obtain a matching cost among two object external contours. This cost is low for similar shapes even if the shape is captured under different rotations or reflections. As shown in Fig. 3, for this task, we first binarize the image. Next, we compute the external contour of the volume by applying a Canny [Can86] edge detector on the binary volume information projected to a particular 2D view. Finally, the contour is vectorized by looking over the contour the distance of each point to the center of mass, see Fig. 3(c). In order to reduce the matching time, we just extract the main external shape silhouette removing inner object contour points from projected 2D shape. We experimentally found that this does not have a negative effect on final performance. Figure 4 depicts the resulting vectorized contour of the foot shown in Fig. 3 where the x-axis corresponds to the position on the contour and the y-axis is the distance of this position to the center of mass. Once the contour is vectorized, it can be efficiently analyzed using dynamic programming. [SK08] presents an indexing approach for time series indexing and mining based on hashing Euclidean distance rules. Although this approach presents promising results for large time series datasets, it is demonstrated that for reduced feature vectors, like the ones we use to codify

the shape silhouette, the estimated warping cost based on dynamic programming uses to obtain more accurate results. In consequence, we use Dynamic Time Warping (DTW). The rationale behind DTW is, given two time series, to stretch or compress them locally in order to make one resemble the other as much as possible. The distance between both time series is computed, after stretching, by summing the distances of individual aligned elements. DTW is an algorithm commonly used for measuring similarity between two sequences which may vary in time or speed. DTW has been applied to video, audio, and graphics -indeed, any data which can be turned into a linear representation can be analyzed with DTW. In this sense, we apply DTW to match the vectorized shapes of each 2D projected volume viewpoint in relation to the reference image.

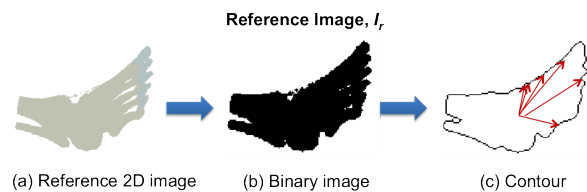


Figure 3: Description of the process for extracting the contour of the reference image.

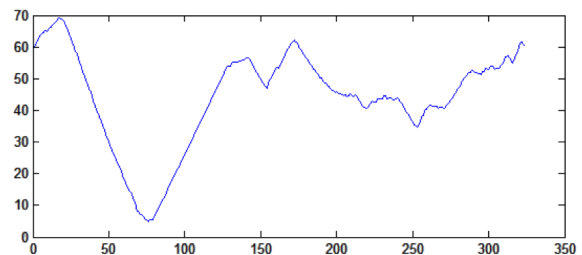


Figure 4: Vectorized contour of the reference image.

In order to obtain reflection invariance we apply the same procedure for the image I_i , and for the horizontal and vertical image reflections, I_i^h and I_i^v , respectively. Figures 5 and 6 depict the process for extracting the contour of a candidate image I_i , and its final vectorized contour, respectively.

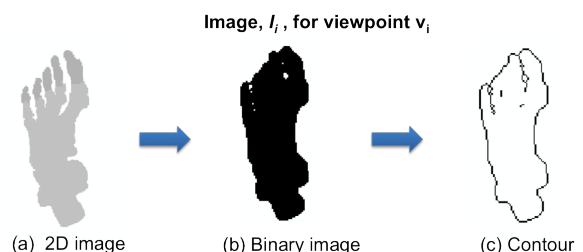


Figure 5: Description of the process for extracting the contour of an image, I_i .

On the other hand, note that rotation and viewpoint invariance will be obtained by the matching procedure

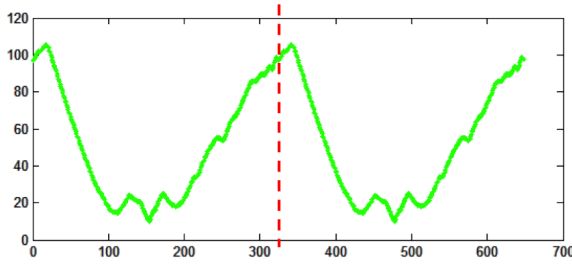


Figure 6: Duplicated vectorized contour of a candidate image I_i to guarantee rotation invariance.

based on Dynamic Time Warping, according to the horizontal and vertical image reflections. Thus, final similarity is obtained as:

$$\text{Dist}(I_r, I_i) = \min(d(I_r, I_i), d(I_r, I_i^h), d(I_r, I_i^v)), \quad (3)$$

where d function computes the Dynamic Time Warping alignment between two series. For this task, I_r is decomposed into a one-dimensional sequence, vectorizing the external shape contour of the connected components in I_r as described above, see Fig. 4. The shape vectors of an image, I_i , are duplicated in size, concatenating twice the vectoring shape to guarantee rotation invariance, see Fig. 6. Then $d(I_r, I_i)$ is computed as a Dynamic Time Warping algorithm. Specifically, our DTW algorithm is defined to match distortions between two models, finding an alignment/warping path between the two shape series I_r and I_i . In order to align these two sequences, a $M_{m \times n}$ matrix is designed, where the position (x, y) of the matrix contains the alignment cost (i.e., defined later in Eq. 5) between positions x and y of both series, see Fig. 7.

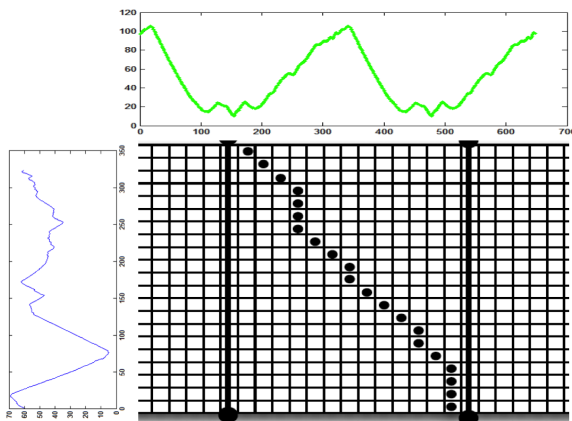


Figure 7: Cost matrix, M , with the cost alignment between positions x and y of both series.

From the matrix, M , a warping path of length T is defined as a set of contiguous matrix elements, defining a mapping between I_r and I_i as $W = \{w_1, \dots, w_T\}$, where w_i indexes a position in the cost matrix. This warping path is typically subjected to several constraints:

Boundary conditions: $w_1 = (1, 1)$ and $w_T = (m, n)$.

Continuity and monotonicity: Given $w_{t-1} = (a', b')$, then $w_t = (a, b)$, $a - a' \leq 1$ and $b - b' \leq 1$, this condition forces the points in W to be monotonically spaced in shape.

We are generally interested in the final warping path that minimizes the warping cost:

$$d(I_r, I_i) = \min \left(\frac{M(w_t)}{|W|} \right), \quad (4)$$

where $|W|$ compensates the different lengths of the warping paths. The cost at a certain position $M(x, y)$ can be found as the composition of the Euclidean distance $c(x, y)$ between the feature vectors of the two sequences and the minimum cost of the adjacent elements of the cost matrix up to that point, i.e.,

$$M(x, y) = c(x, y) + \min(M(x-1, y-1), M(x-1, y), M(x, y-1)). \quad (5)$$

The cost $c(x, y) = (x - y)^2$ that is the Euclidean distance between the feature vectors at positions x and y .

In order to detect the beginning and ending positions of the candidate shape that minimizes the matching cost, the current ending cost is checked (the cost of the element in the last row). This minimum value is assigned to $d(I_r, I_i)$, which is used in Eq. 3. Figure 8 depicts in green color the best alignment (i.e., the one that minimizes the matching cost) among the vectorized shapes of the reference image, I_r , and the candidate image, I_i .

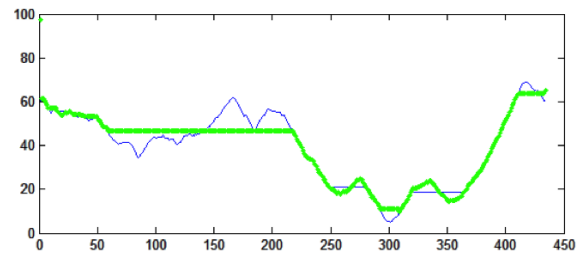


Figure 8: Alignment found using DTW among I_r and the candidate image I_i .

Given that some labeled structures may have similar shape but different resolution, and thus offer different percentage of information, we complement the DTW similarity procedure with a correlation of information from each feature f_j , $f_j \in F$, among two images. The features we use in our foot example are the labels of the volume (i.e., toes, palm and ankle). For this task $Corr$ is computed as follows:

$$\text{Corr}(I_r, I_i) = \sum_{f_j \in F} \left| \frac{\text{count}(I_r, f_j)}{\sum_{f_i \in F} \text{count}(I_r, f_i)} - \frac{\text{count}(I_i, f_j)}{\sum_{f_i \in F} \text{count}(I_i, f_i)} \right|, \quad (6)$$

where $\text{count}(I_r, f_j)$ estimates the number of visible pixels of feature f_j in I_r . When transparency is taken into account in the rendering, each projected pixel can contribute several times in the count function for different structure labels, as many as non-occluded structures

have been projected onto that pixel. Thus, in Eq. 6 normalization is based on the total amount of visible pixels for each structure. In this sense, similar external shape projections but with different amount of visible information from each structure in relation to the reference image will penalize the similarity measure. On the other hand, similar visual proportion for each structure in comparison with the reference image will be penalized if the external projected structure among views differs. This trade-off regularized by the β term in Eq. 2 allows for a robust estimation of complementary view selection, assigning higher score to those views that offer additional information to the primary view in terms of visual structures and external shape.

3.3 Selecting the optimal complementary camera

Given a reference camera viewpoint $v_r \in S$ and its projected image I_r , we define the function $Dual(v_r)$ as¹,

$$Dual(v_r) = \max(Sim(I_r, I_i) \cdot H(v_i)) : \forall I_i \in I. \quad (7)$$

Similarity measure defined in Eq. 2 is used to mask the entropy at each sampled viewpoint location v_r into the sampled space S .

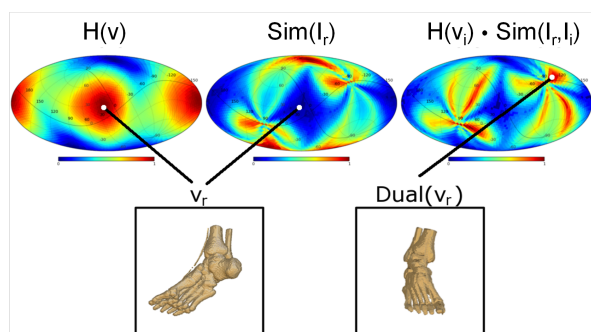


Figure 9: The first row shows the projection of the sampled sphere of each computed viewpoint function. The second row shows the reference image and the complementary camera projection. The entropy $H(v)$, similarity $Sim(I_r, I_i)$ and the final composition $H(v_i) \cdot Sim(I_r, I_i)$ is used to find the complementary viewpoint, $Dual(v_r)$.

Figure 9 describes the final composition to find the optimal complementary viewpoint. The entropy $H(v)$ is computed for all the selected features (or isovalues) (see Eq. 1). Each projected image stores for each pixel the first intersection with the focused feature and the corresponding depth, when this intersection exists. These images are next used to compute the entropy, $H(v)$. Next, given a viewpoint location, v_r , of the sampled sphere, S , the similarity of all the samples related can

be computed using the $Sim(I_r, I_i)$ function. Finally, the maximum value of these former costs locates the complementary camera $Dual(v_r)$.

4 SIMULATIONS

This section describes the design of the experiments, the analysis of the proposal for computing complementary camera viewpoints and the obtained results.

4.1 Design of the experiments

We have analyzed the proposed method using an Intel Core I7 870 with 16 GB memory equipped with an NVidia GeForce GTX 690 GPU with 4GB of GDDR5 memory. The viewport size is 512×512 .

In order to test our method, we have sampled the searching space S with 192, 768 and 3072 viewpoint locations. We are able to compute 192 views in 0.03 seconds in average, 768 in 0.15 seconds and 3072 in 0.61 seconds. We have used three data sets: (1) the *Foot*² data set is a 128^3 -sized CT scan of a human foot; (2) the *Thorax*³ data set represents a phantom human torso of 400^3 voxels; and (3) the *Walnut*⁴ data set, which is a CT scan of a walnut of $400 \times 296 \times 352$ voxels. In the walnut dataset, the structures have a configuration of onion-peel-like, i.e., each structure is totally or partially contained in another structure.

We have selected three structures of the foot (toes, palm and ankle), twelve for the thorax to test different amount of overlapping between structures, and three features for the walnut (i.e., shell, seed, and core segments). The selected structures are based on the public label annotations of the used datasets. We empirically setup the β parameter to 0.5 in order to compute the similarity term.

With the aim of validating the goodness of our proposal, we performed a live-user evaluation. In our trial participated 20 subjects. The users were asked to score a best view and a set of best complementary views in relation to the reference ones. With this experiment we look for the correlation among observers to the best and complementary viewpoint selection based on the mean number of votes per viewpoint. This 3D map serves as a qualitative evaluation in order to look for observers agreement. We also use this information to compare the user selection with our automatic generated 3D score maps for the best and the complementary viewpoints.

4.2 Analysis

In our analysis, the trial performed by the users consisted of the four following steps. Firstly, each trialist rated all the images in the range [1..5]. Secondly,

¹ We experimentally found that the product rule obtained better trade off results in comparison to other metrics, such as the sum rule.

² <http://www.slicer.org/archives>.

³ <http://www.voreen.org>.

⁴ <http://www.uni-muenster.de/EIMI/>.

among all the images, each trialist selected the one that he considered best reference image (I_r). Next, looking at this reference image and the remaining ones again, they were asked to rate them according to how good the complementary view each one offers is. Finally, they were asked to locate the best complementary image to the one previously chosen as the reference image. For the sake of simplicity, all users were asked to rate the best viewpoint only on the Thorax data set, meanwhile they were asked to rate all the complementary viewpoints for the three data sets. For all the cases of study, 192 images were provided to the participants.

On the other hand, we did similar steps for evaluating our proposal. First of all, for each viewpoint, we rated them using the entropy, $H(v)$. Secondly, we selected as reference image, I_r , the viewpoint with the maximum entropy. Next, we computed the shape similarity term. Finally, we selected the optimal complementary camera for the previously selected reference image, I_r , as described in Section 3.3.

Table 1 depicts the results for the Thorax data set with 192 viewpoint locations for the live-user evaluation and the proposed algorithm. The first observation for the best camera selection on the Thorax data set is that the users tend to rate all those viewpoints that they do not consider relevant very low in the scale, meanwhile they offer high values to those viewpoints they consider relevant. In this sense the densities of the 3D map tend to be a bimodal distribution. In contrast, in our case our automatically computed map tends to be a Gaussian distribution of scores. The important point of this experiment is that there exists a high agreement among observers in relation with the best viewpoint and that this viewpoint (Table 1 (a) and (e)) highly correlates with the one obtained by our automatic method (Table 1 (b) and (f)). Moreover, both distributions of scores of the best view share similar patterns.

In relation to the complementary viewpoints, as depicted in the figures of Table 1 (c) and (d), there also exists a high agreement among observers and a high correlation with our computed scores for complementary viewpoint selection. It is remarkable that for some symmetric volumes, such as the Thorax data set, we still obtain a high score for that view (Table 1 (g) and (h)), although the average complementary view chosen by the users do not correspond to the one selected by our system. This is mainly because we obtain similar scores for opposite viewpoints given the symmetry of the visualized objects, especially when the number of analyzed viewpoints tends to be 3072 views.

In Table 2, we also show the best complementary selection obtained by the live-user evaluation for the Foot and Walnut data sets based on our best primary view selection. Comparing these views with the complemen-

tary ones computed by our approach (see Table 3 and 5) one can see a high agreement among the selected views.

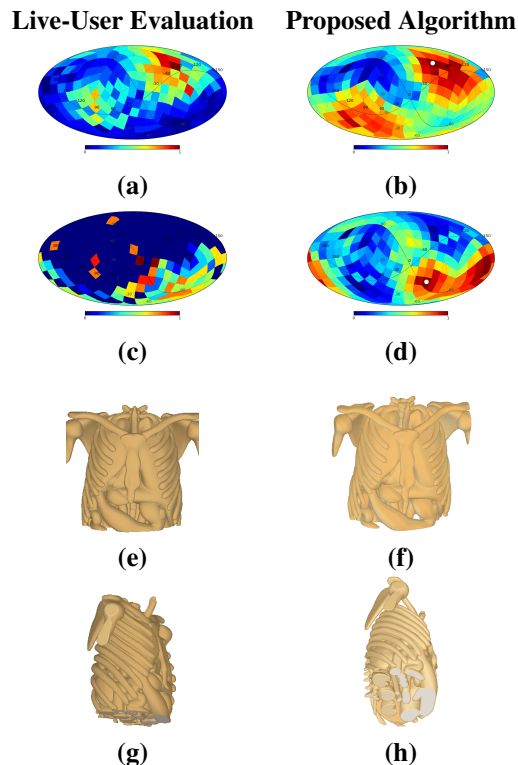


Table 1: Comparison of the results obtained with the live-user evaluation and our proposed method for the Thorax data set with 192 viewpoint locations: (a) and (b) show the initial score distribution, (c) and (d) refer to the complementary view score distribution, (e) and (f) show renderings of the reference image I_r selected by users and computed by our proposed algorithm, respectively, and (g) and (h) show renderings of the complementary view in relation to the same reference image.

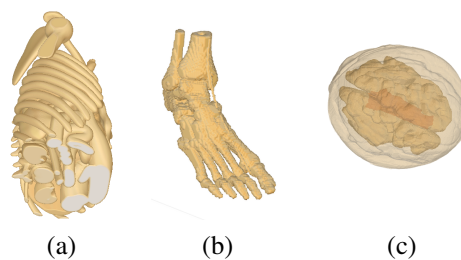


Table 2: Best complementary camera selection by live-user evaluation on the (a) thorax, (b) foot and (c) walnut data sets.

In the tested data sets, the convergence of the complementary camera location directly depends on the variation of the number of samples on the bounding sphere as well as the used searching space.

Tables 3, 4, and 5 show the obtained results for different number of samples in the sphere, $|V|$, from 192 to 3072. The first three rows represent the $H(v)$ entropy of

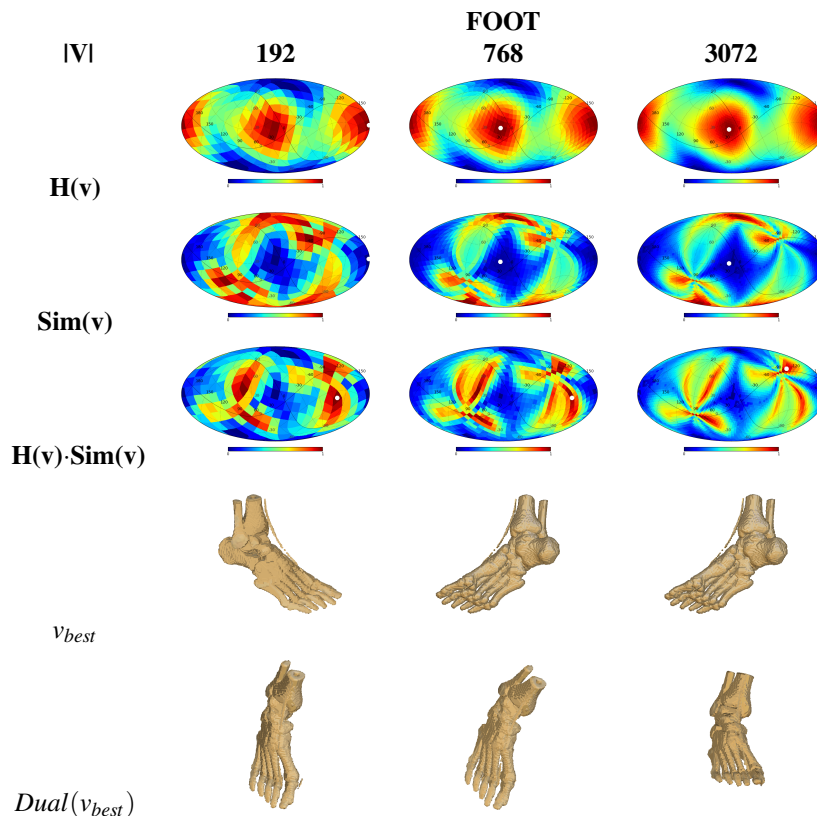


Table 3: A comparison with different number of sampled viewpoints for Foot data set.

the sampled viewpoints over the sphere, the $Sim(v)$ associated the concrete view, and the $H(v) \cdot Sim(v)$ composition, respectively. Each of them is mapped to a plot that represents the whole set of sampled viewpoints in the sphere S . The best viewpoint location in the first row (v_{best}) is highlighted with a white bullet. The similarity plot is computed according to this location ($Sim(v_{best})$) in the second row. Finally, the third row shows the $Dual(v_{best})$ as a white bullet in the composed plot of $H(v) \cdot Sim(v)$. The two last rows are the final renderings associated to v_{best} and $Dual(v_{best})$, respectively.

It can be observed that the computed entropy maps become stable between 192 and 768 cameras. Similarity maps converge from 768 to 3072 sampled positions. Additionally, if we evaluate the final renderings, the 768 is enough in order to obtain a dual camera position with complementary information to the first camera. In this experiment, the reference camera is the best viewpoint located in the entropy map, but we can define any arbitrary position, and then compute the similarity map and its dual function. Note that the complementary camera viewpoints computed highly correlate on average to the ones defined by the live-user evaluation validation.

In addition, we tested the accuracy of the similarity in transparent scenes. Walnut data set is composed of onion-peel-like structures. We analyzed the number of visible pixels of each feature taking into account that several structures can be projected in a pixel. The

count function introduced in Section 3.2 allows one to compute the similarity of this kind of projection. Our method maintains the convergence of the complementary camera between 192 and 768 views for opaque regions as well as the non-transparent structures.

5 CONCLUSIONS AND FUTURE WORK

Traditional research in volume visualization has been focused on involving the user to assist on gathering visual information contained in the data. With this approach users require some expertise, apart from being introduced in a difficult and tedious task in order to setup the parameters of the visualization. In this paper we have described a proposal for obtaining a representative complementary camera in volume rendering without user intervention. Moreover, the proposal is easy to evaluate and robust to changes in the resolution of the final view. Specifically, the proposed method is able to cope with the problem of locating automatically the complementary camera that maximizes the complementary information of the structures. It combines the quantity of information each camera provides for each structure and the shape similarity of the projections of the remaining viewpoints based on Dynamic Time Warping. We have demonstrated the efficacy of our proposal against a benchmark of three data sets. In general, our experiments indicate that the proposal has the po-

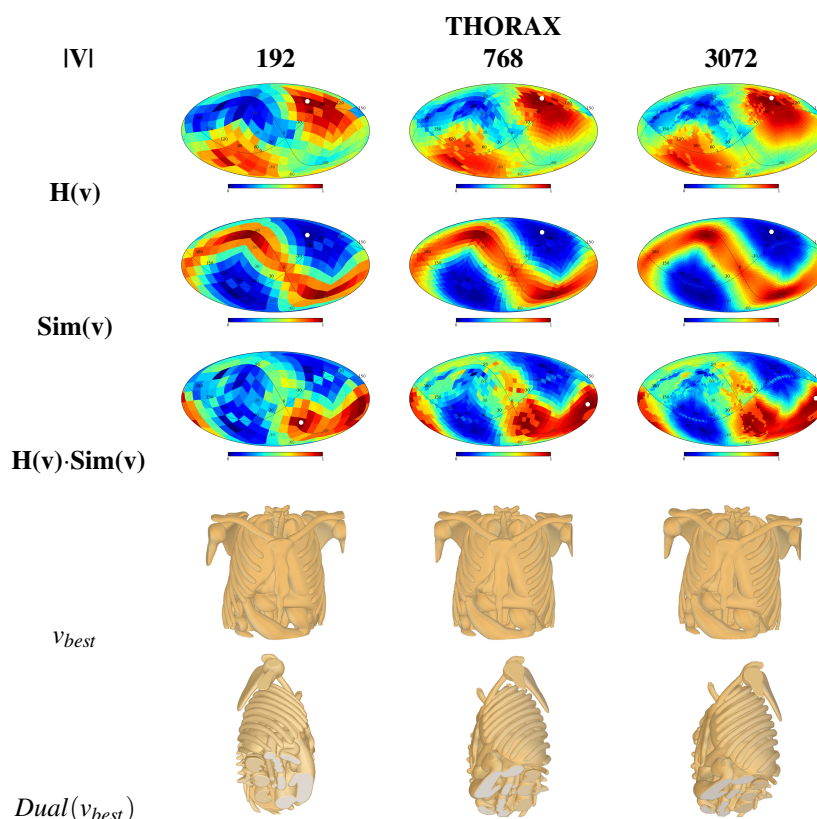


Table 4: A comparison with different number of sampled viewpoints for Thorax data set.

tential to deliver worthwhile similarities in comparison with real users. Our live-user analysis has shown that our approach is close to the preferences of the users. Our results also show that the proposal quickly converges and, with only a 768 sampled positions, it is able to obtain a dual camera position with complementary information to the reference camera. Our future work will focus on exploring new view-dependent descriptors. Currently, we are working on the parallelization of the whole process using OpenCL to get real-time.

ACKNOWLEDGEMENTS

Work partially funded by TIN2011-24220 and MICINN Grant TIN2009-14404-C02 Spanish research projects.

6 REFERENCES

- [BG05] BRUCKNER S., GRÖLLER E.: Volumeshop: An interactive system for direct volume illustration. In *IEEE Visualization 2005* (Oct. 2005), pp. 671–678.
- [BS05a] BORDOLOI U., SHEN H.: View selection for volume rendering. In *IEEE Visualization* (2005), p. 62.
- [BS05b] BORDOLOI U., SHEN H.-W.: View selection for volume rendering. In *IEEE Visualization* (2005), p. 62.
- [Can86] CANNY J.: A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 8, 6 (June 1986), 679–698.
- [CMH08] CHAN M. Y., MAK W., H.QU: An efficient quality-based camera path planning method for volume exploration. In *Proc. ISVC 2008, LNCS-5359* (2008), et al (eds.) G. B., (Ed.), Springer-Verlag Berlin Heidelberg, pp. 12–21.
- [CQWZ06] CHAN M. Y., QU H., WU Y., ZHOU H.: Viewpoint selection for angiographic volume. In *Proc. ISVC 2006, LNCS-4291* (2006), Springer-Verlag, pp. 528–537.
- [GHB*05] GORSKI K., HIVON E., BANDAY A., WANDEL T B., HANSEN F., REINECKE M., BARTELMAN M.: Healpix: a framework for high resolution discretization, and fast analysis of data distributed on the sphere. *The Astrophysical Journal* (2005), 622–759.
- [GP09] GRAU S., PUIG A.: An adaptive cut-away with volume context preservation. In *Advances in Visual Computing* (2009), vol. 5876 of LNCS, pp. 847–856.
- [KBKG08] KOHLMANN P., BRUCKNER S., KANITSAR A., GRÖLLER E.: The livesync++: Enhancements of an interaction metaphor. In *Graphics Interface*

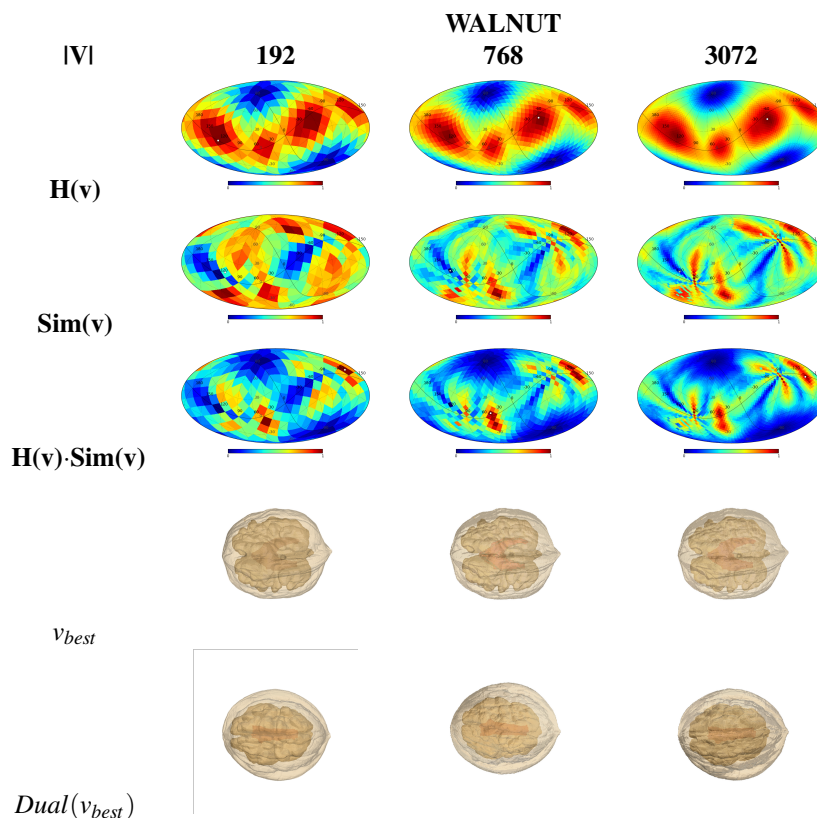


Table 5: A comparison with different number of sampled viewpoints for the Walnut data set.

- (08), pp. 81–88.
- [KBKG08] KOHLMANN P., BRUCKNER S., KANITSAR A., GRÖLLER M.: The livesync++: Enhancements of an interaction metaphor. In *Graphics Interface* (2008), pp. 81–88.
- [MNTP07a] MÜHLER K., NEUGEBAUER M., TIETJEN C., PREIM B.: Viewpoint selection for intervention planning. In *IEEE Symp. on Visualization 2007* (2007), pp. 267–274.
- [MNTP07b] MÜHLER K., NEUGEBAUER M., TIETJEN C., PREIM B.: Viewpoint selection for intervention planning. In *Proceedings of Eurographics/IEEE VGTC Symposium on Visualization 2007* (2007), pp. 267–274.
- [PPB*05] POLONSKY O., PATANE G., BIASOTTI S., GOTSMAN C., SPAGNUOLO M.: What’s in an image? *The Visual Computer* 21 (2005), 840–847.
- [PPVN08] P. P. VÁZQUEZ E. M., NAVAZO I.: Representative views and paths for volume models. In *Smart Graphics* (2008), pp. 106–117.
- [RBGV08] RAUTEK P., BRUCKNER S., GRÖLLER M. E., VIOLA I.: Illustrative visualization: New technology or useless tautology? *SIGGRAPH Comput. Graph.* 42, 3 (2008).
- [SK08] SHIEH J., KEOGH E.: isax: Indexing and mining terabyte sized time series, sigkdd, pp, 2008.
- [TFTN05] TAKAHASHI S., FUJISHIRO I., TAKESHIMA Y., NISHITA T.: A feature-driven approach to locating optimal viewpoints for volume visualization. *IEEE Visualization 0* (2005), 63.
- [VFSG06] VIOLA I., FEIXAS M., SBERT M., GRÖLLER E.: Importance-driven focus of attention. *IEEE Trans. on Visualization and Computer Graphics* 12, 5 (2006), 933–940.
- [VKG05] VIOLA I., KANITSAR A., GRÖLLER E.: Importance-driven feature enhancement in volume visualization. *IEEE Trans. on Visualization and CG* 11, 4 (2005), 408–418.
- [ZWD12] ZHANG Y.-S., WANG B., DAI C.-J.: Viewpoint Selection Based on NM-PSO for Volume Rendering. In *Intel. Comp. Theories and Applications*, vol. 7390. 2012, pp. 487–494.

Cartographic Rendering of Elevation Surfaces using Procedural Contour Lines

Neha Manyà

Louisiana State University
Center for Computation &
Technology

Department of Computer Science
and Engineering Division

3127 P. F Taylor Hall

USA, LA-70803, Louisiana

nmanyà1@tigers.lsu.edu

Isaac Ayyala

Louisiana State University
Center for Computation &
Technology

Department of Electrical and
Computer Engineering

3101 P. F. Taylor Hall

USA, LA-70803, Louisiana

iayyal1@tigers.lsu.edu

Werner Benger

Louisiana State University
Center for Computation &
Technology

213 Johnston Hall
USA, LA-70803, Louisiana

werner@cct.lsu.edu

ABSTRACT

We describe a method to render a height field given on a triangular mesh in a manner that is similar to well-known cartographic map views. This approach utilizes hypsometric tints and creates topographic contour lines procedurally as part of the rendering process. We discuss essential graphical enhancement techniques such as line smoothing, gradient compensation, contour line visibility fading, nonlinear elevation mapping, and shininess variation. The method is exemplified upon various numerical data sets from application sciences, demonstrating its ability to visually allow quantitative and qualitative precise assessment of the data values.

Keywords

Contour lines, hypsometric tints, scientific visualization, bathymetry, coastal modeling, terrain visualization.

1. INTRODUCTION

The challenge of rendering bathymetric data is emphasizing features over a large scale while sustaining accurate data representation. Cartography is an ancient field of interest to draw upon for this purpose. The use of colors to represent ranges of elevation, known as hypsometric tints, is said to have been invented by Leonardo da Vinci [Vin03]. Contour lines complement hypsometric tints to depict more detailed information quantitatively. Both methods are familiar and intuitive. A similar approach is therefore highly desirable for application upon numerical datasets where interactive exploration is mandatory for data analysis, especially when exploring complex geomorphology.

Deltaic estuaries in the northern Gulf of Mexico are good examples of complex geomorphological systems. These estuaries are characterized by numerous channels and wetland networks on the order of 20m – 100m in size, and have a highly complex bathymetry within a fairly narrow depth

range of 0m – 5m. Implementation of 3-dimensional hydrodynamic models is particularly challenging in these systems because numerical grids need to resolve both the complex bathymetry and intricate wetland features. Consequently, bathymetric rendering and visualization can provide invaluable information for the generation of 3D numerical grids.

Related Work

The Collaborative Ocean Visualization Environment (COVE [Gro10]) has been designed with oceanographers to provide a set of elaborated tools for bathymetric data exploration, including hypsometric tinting and display of simple contour lines. In our approach we seek to achieve even higher accuracy in emphasis of finer variations in elevation, limited only by numerical precision. In [Gul10] an algorithm is presented to extract contour lines from scanned color topographic maps. Topography maps already have the contour lines in them, but they might be broken or overlapping. They provide techniques to filter noise, remove bifurcations and connect lines properly. In contrast, here we intend to create contour lines from the data itself. [Du04] discusses the limitations of contour line generation. They found that useful contour lines need to be joined, but should only be extracted if they were not too fragmented and enough spatial information was available. The quality of procedural contour lines such as in our approach will of course depend on the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

quality of available spatial information as well, but joining and fragmentation of lines is not a hindrance.

2. RENDERING TECHNIQUE

We intend to render a height field, such as given in the form of a triangular surface mesh, similar to the well-known cartographic views known from topographic maps. Firstly, this implies using a colorization scheme widely known as hypsometric tints: it renders bathymetric regions (below ‘sea level’) using bluish shades and hypsometric regions (above sea level) in shades from green via yellow to brown to red and finally gray – white colors, in approximation of terrestrial average colors. Secondly, we like to draw contour lines to allow quantitative reading of height levels. Because explicit calculation of contour lines as required for numerically precise analysis is computationally expensive (even when optimized using marching squares as 2D version of marching cubes [Lor87]), here we want to provide a quick preview method by implementing an approximation of contour lines as part of the surface shading process. A lookup table (LUT) containing ten entries for bathymetric and ten entries for hypsometric color values is sufficient to provide a coarse overview of the data set. The intended purpose of contour lines is to complement the colorization by providing more detailed elevation information. While they may be implemented via a very large, high-resolution LUT, the availability of programmable GPUs allows implementing them procedurally. Thus, the achievable resolution is only limited by numerical precision. The basic idea is to make use of the modulo function $h \bmod \lambda := h - \text{floor}(h/\lambda)$ for a given height h . Depending on a level height (e.g. 1m, 10m, 100m) and a given finite thickness δ of the contour line, it determines whether the hypsometric tint from the LUT or a constant color for the contour line, e.g. black or red, is used:

$$\text{color}(h) = \begin{cases} \text{Contour}, & h \bmod \lambda < \delta \\ \text{LUT}(h), & h \bmod \lambda \geq \delta \end{cases}$$

Using this modulo function it is straightforward to implement multiple such contour lines at the same time using different parameters. A set of five types of contour lines is easy to interpret visually as it is familiar from cartographic displays. These may be placed as red thick lines at levels of 10 (0m, 10m, 20m, ...), thick black lines shifted by 5 (5m, 15m, 25m, ...), red thin lines at levels of 1 (1m, 2m, 3m, 4m, 6m, 7m, ..), black thin lines shifted by 0.5 (0.5m, 1.5m, 2.5m, 3.5m, 4.5m, ...) and finally gray lines at distance levels of 0.1 (0.1m, 0.2m, 0.3m, 0.4m, ...). This scheme allows reading off a height range of 1:100 quickly and even 1:1000 if we consider a line width of 1cm, as demonstrated in Figure 1. While this approach is pretty straightforward, there are several aspects that can be, or need to be improved.

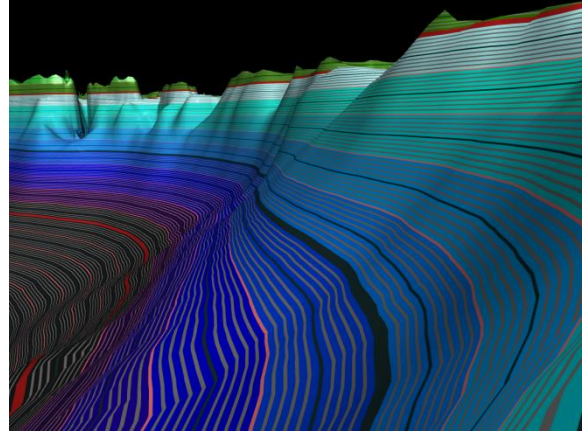


Figure 1: Contour lines as integrated part of surface colorization, utilizing five categories of contour levels.

Line Smoothing

A direct implementation of the modulo function leads to strong aliasing (moiré) effects, which disturbs the visual appearance since for each pixel there will be a binary yes/no decision on whether it belongs to a contour line or not. We can address this problem by replacing the binary decision $\text{color}(h)$ with a smooth transition between the contour line and the ground color by using a hermite spline interpolation as in

$$S(x_0, x_1, x) = \begin{cases} 0, & x \leq x_0 \\ t^2(3 - 2t), & x_0 < x < x_1 \\ 1, & x \geq x_1 \end{cases}$$

where $t = (x - x_0) / (x_1 - x_0)$. Such a function is provided by the GLSL smoothstep() call.

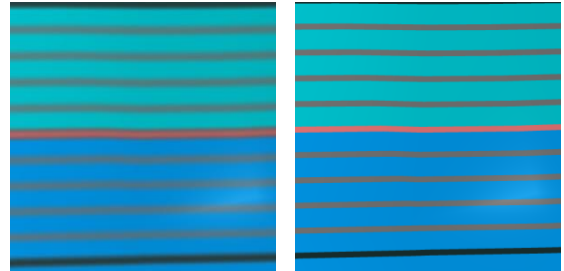


Figure 2: Line smoothing to address aliasing.

We then introduce a smoothing distance ε , which can be set to be $\varepsilon = \delta/4$, for instance, and compute a blending weight $W = B_s T_s$ as product of a bottom weight B_s and top weight T_s . The bottom part of the contour line is calculated as

$$B_s = S(0, \varepsilon, (h + \delta) \bmod \lambda) .$$

The top part is calculated via

$$T_s = (1 - S(\delta - \varepsilon/2, \delta + \varepsilon/2, (h + \delta) \bmod \lambda)) .$$

Both functions smooth the line’s edges by ε . We need to consider that the line shall be centered precisely at height h , extending by δ upwards and

downwards, smoothing out over a distance ε . The blending weight W is 1 for h being an integer multiple of λ such that the final color is given as

$$color(h) = (1-W) \cdot LUT(h) + W \cdot Contour .$$

Therefore, by using the above formulae, the edges of the contour lines are smoothed, implementing antialiasing.

Gradient Compensation

As contour lines are implemented in this approach by identifying a set of points within the same height range, we will naturally find more points fitting this criterion in shallow regions, resulting in a broadening of the lines depending on the terrain's slope. The problem is inherent by design because a contour line becomes undefined in a completely flat area. While this broadening of lines is visually displeasing as compared to the computation of an explicit contour line, it may also be considered as a depiction of the contour lines becoming numerically unstable in flat areas. This would not necessarily be evident from an explicit contour line. Therefore, this effect cannot be fully cured. But, it can be compensated to some extent by taking into account the slope of the terrain and introducing a dependency of the contour line's thickness δ on the surface normal vector n ($n_z=1$) via

$$\delta = \delta_0 (1 - n_z^2)^g .$$

Here, g is an exponent allowing fine-tuning the 'gradient compensation'. "Mathematically correct" would be an exponential of 0.5, however in practice the ability of over-compensation turns out to be beneficial, as depicted in Figure 3.

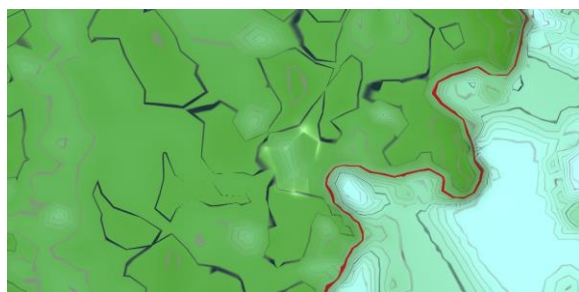
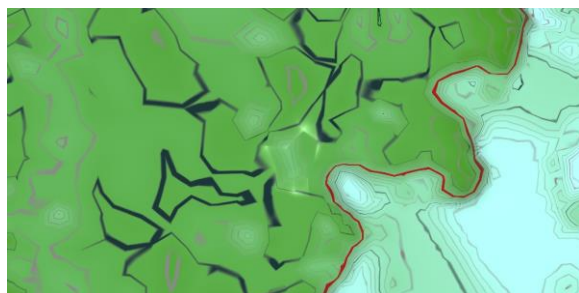


Figure 3 Procedural contour lines in a shallow wetland area, as-is and using a 'gradient compensation' exponent of 16 to narrow lines.

Visibility Fading

Contour lines of constant thickness as in Figure 4 come with the disadvantage that a thickness value working pleasingly well in the foreground leads to invisible or strongly aliased lines at further distances. In turn, a thickness value providing good background appearance looks ugly in the foreground. We thus relate both the blending weight W as well as the contour line thickness δ to the apparent size of the contour line on the screen, such to have lines in the distance being broader, but less prominent, and lines close to the observer being smaller but crisper. However, we cannot simply introduce a dependency of the surface point on the observer's location because this would fail in the case of an orthographic camera projection. Instead, we need to compute a visibility measure that is related directly to the apparent size of the contour line on the screen.

We proceed as follows: given the normal vector n of the surface, we can get a vector t tangential to the contour line by taking the cross product with the 'up' vector $z=(0,0,1)$ as $t = n \times z$. The tangential vector t will always be parallel to the horizontal plane, i.e. $t_z=0$. However, this expression also applies to a general underground, such as the curvature of Earth, by using an arbitrary local 'up'-vector z . From this tangential vector we compute a vector s parallel to the surface but orthogonal to the tangent via $s = n \times t$. Now, given a point on the surface P we can compute the location Q of the upper boundary of our contour line as $Q = P + \delta s$. Both Q and P , given in world coordinates, are then projected in screen coordinates by multiplying in 4D with the OpenGL projection and modelview matrix M , yielding 4D homogenous screen coordinates ${}^sP=M P$, ${}^sQ=M Q$. 3D affine screen coordinates are computed via dividing by the 4th component: ${}^aP=({}^sP_x/{}^sP_w, {}^sP_y/{}^sP_w, {}^sP_z/{}^sP_w)$, ${}^aQ=({}^sQ_x/{}^sQ_w, {}^sQ_y/{}^sQ_w, {}^sQ_z/{}^sQ_w)$. We are only interested in the x and y components of these projected points as they are directly related to the size of the contour line in pixels. To get the actual number of pixels we would need to multiply x and y with the width and height of the viewport in pixels. However, we want a pixel-resolution independent algorithm and thus only consider normalized screen coordinates. We define a visibility factor V as

$$V = V_0 \sqrt{({}^aP_x - {}^aQ_x)^2 + ({}^aP_y - {}^aQ_y)^2} ,$$

based on some adjustment factor V_0 to fine-tune the visual appearance. The visibility factor will be small at large distances of the contour lines from the screen as well as the contour-line being viewed tangentially, which would result in a visibly small line as well. To compensate for aliasing effects and enhance distant lines, we increase the line width via dividing by the

visibility factor while at the same time reducing its blending weight:

$$\delta_{eff} = \delta / V \quad , \quad W_{eff} = W \cdot V \quad .$$

Using this effective line thickness δ_{eff} and blending factor W_{eff} , we achieve visually pleasing crisp lines in the foreground, fading out into bigger lines emphasizing coarser structures in the background, as demonstrated vs. Figure 5.

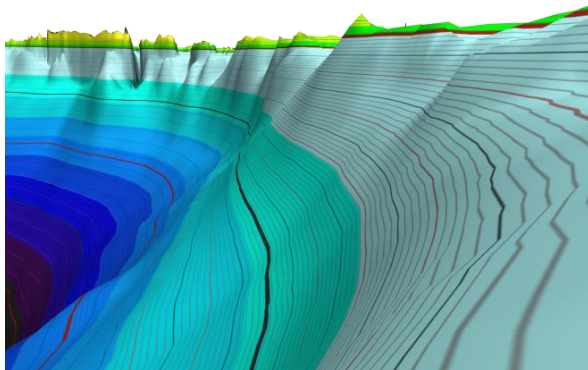


Figure 4 Contour lines with no distance dependency, leading to overly thick foreground lines while distant lines remain hardly visible.

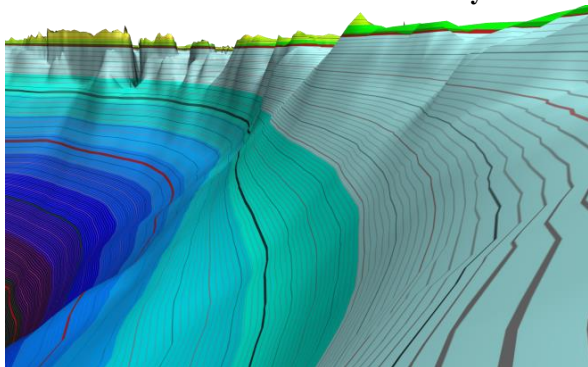


Figure 5 Contour lines with visibility fading, enhancing details crisper in the foreground as well as in background.

Non-linear Hypsometric Tints

Classically each entry in the color lookup table covers the same height range to allow easy identification of elevation by its color. A non-equidistant mapping in contrast provides the advantage of allowing emphasizing finer variations in elevation while covering the same height range. Whereas, the combination with contour lines still allows quantitatively precise read-off. One pretty simple choice of a non-linear mapping of height to colors is to utilize the arctangent function, as it maps an infinite range $[-\infty, +\infty]$ of input values to the range $[-\pi/2, +\pi/2]$. Mapping this range to the interval $[0,1]$ for indexing a LUT is straightforward. We can further introduce a power function to parameterize the mapping of height h for additional adjustment:

$$C_l = \frac{1}{2} \left[\left(\frac{\arctan(h)}{\pi/2} \right)^\gamma + 1 \right]$$

where $C_l \in [0,1]$ is the LUT indexing value and $0 < \gamma < \infty$ is a tunable exponent. Values smaller than 1 will compress color equipotential lines around the sea level $h=0$ and enhance low-elevation details, as depicted in Figure 1.

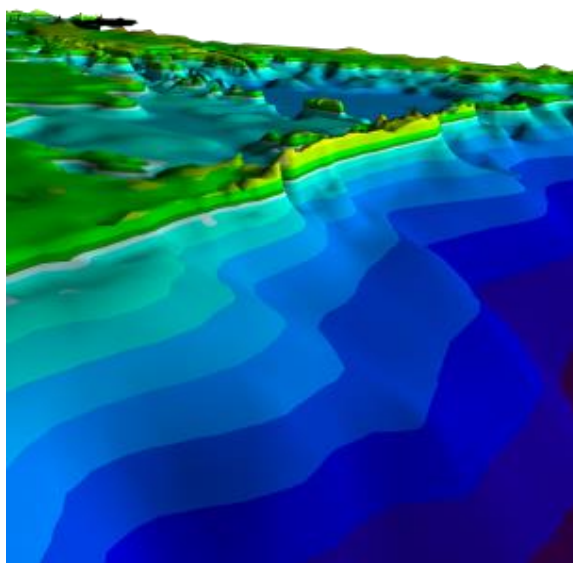
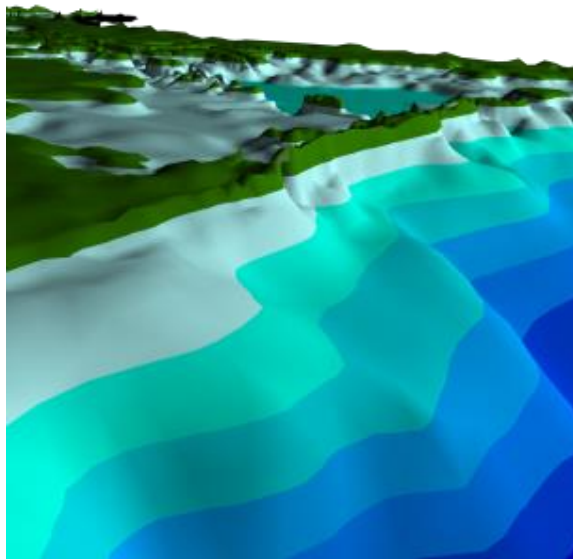


Figure 6 : Linear and non-linear colorization.

Shininess Variations

Modification of the shininess parameter as part Phong shading allows to view the surface and the below the sea level of the grid differently.

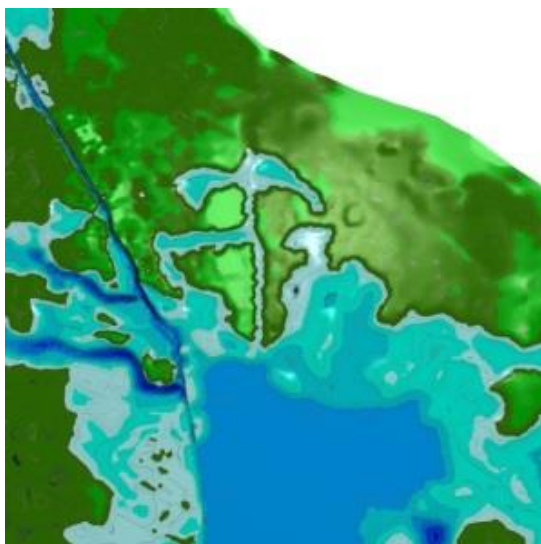
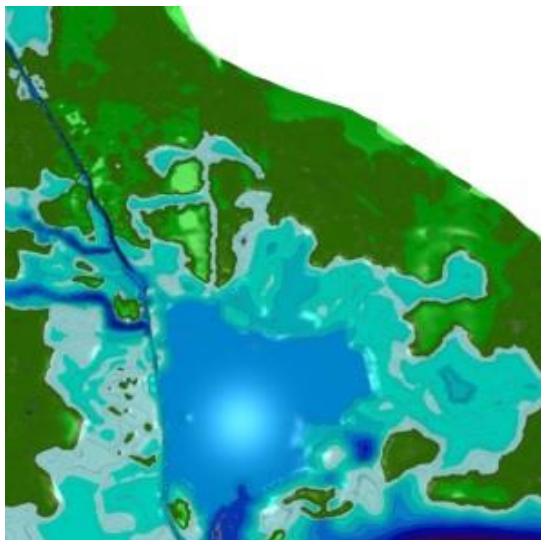


Figure 7: Variation of surface reflectivity (shininess) applied to water bodies within a shallow wetland area.

3. APPLICATION

Icosahedron

We use an icosahedron as a very simple example of a triangular mesh to validate aspects of our technique. Here, the height values range from -1.618 to +1.618. Taking this range into consideration, a set of contour lines is shown. Basically, three red thick lines are shown representing elevation levels of -1, 0, +1.

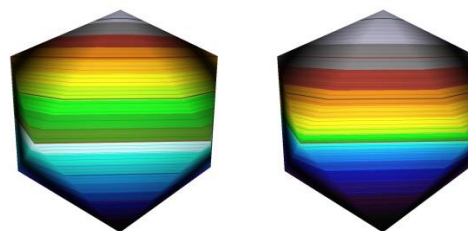


Figure 8: Linear and non-linear colorization without smoothing effect.

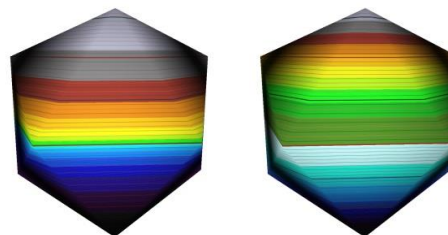


Figure 9: Non-Linear colorization with power less than one and greater than one

Barataria Bay

Barataria Bay is a 120 km long estuary located in the north-central Gulf of Mexico, just to the west of the Mississippi River Delta (Figure 10). It is bounded by the Mississippi River to the east and by a former channel of the Mississippi River, Bayou Lafourche, to the west. The estuary is connected to the Gulf of Mexico through four tidal passes and contains several large lakes and numerous marshes interconnected by ponds and channels. The average depth is only about 2 m but there are several deeper channels (e.g., ~ 4 m deep Barataria Bay Waterway) and tidal inlets (e.g., ~ 20 m deep Barataria Pass).

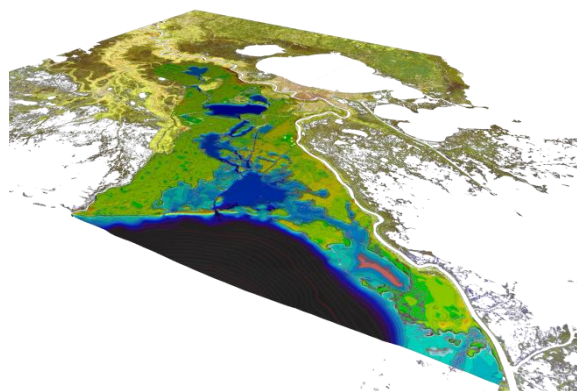


Figure 10 Barataria Bay estuary: Mississippi river is at right, New Orleans just above the center.

Here we explore a triangular mesh consisting out of 63190 vertices and 125038 triangle elements, covering a total area of 3140km², out of which 2950km² is open water. The grid resolution ranges from 18m in the channels to 1600m over wetlands and open water bodies. This mesh carrying precise

bathymetric information is the basis for several numerical hydrodynamic models [Ino08, Das10, Das11, Das12].

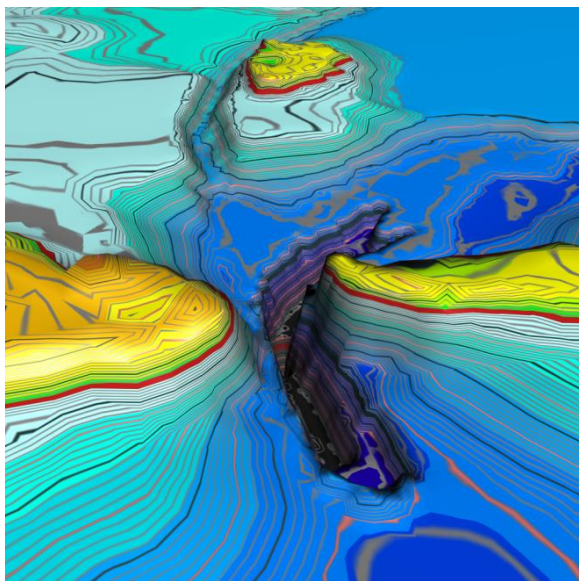


Figure 11 Barataria Pass

The maximum and minimum elevations are at 8.2m and -22.4m, respectively. Our contour line algorithm automatically places contour lines at the most appropriate power of 10 levels, therefore placing three thick red lines at levels of 0.0m, -10m and -20m. Thin red lines are placed at 1m levels, alternating with thin black lines offset by 0.5m. Gray lines represent elevation levels of 10cm.

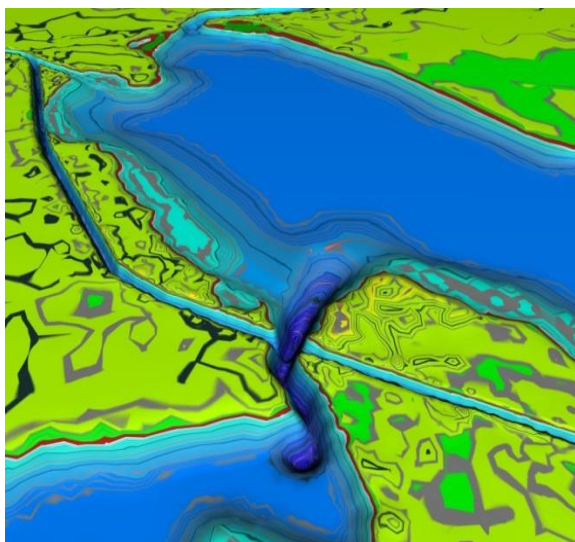


Figure 12 Lake Salvador outlet

Important features in this numerical mesh such as Barataria Pass, Figure 11, or Lake Salvador outlet, Figure 12, are clearly emphasized when applying our cartographic 3D rendering. With proper vertical scaling, Barataria Bay Waterway that carries a large portion of the water flow in the area is represented

more prominently when compared with a conventional 2D cartographic view.

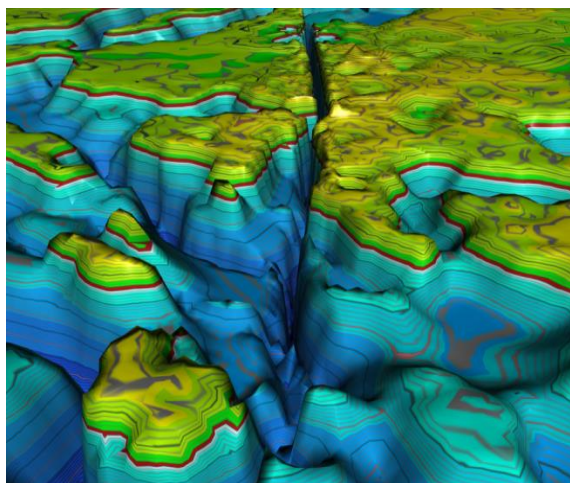


Figure 13 "Grand Canyon of Barataria Bay" - Barataria Bay Waterway.

Optionally, we may also just display the contour lines on their own by rendering the surface itself transparent but only the contour lines opaque. This effect on the Barataria Bay waterway is demonstrated in Figure 14, resulting in a fine detailed depiction of contour levels that are automatically adjusting to view distance.

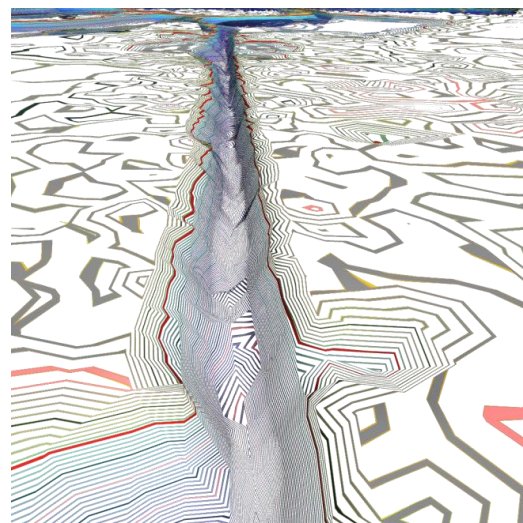


Figure 14 Barataria Bay Waterway displaying procedural contour lines as derived from the numerical triangular mesh.

Rheinfelden LIDAR Terrain

The "Rheinfelden" data set has been derived from an airborne light detecting and ranging (LIDAR) laser scan using a bathymetric green laser system, the RiegI hydro-graphic laser scanner VQ-820G [Ste10]. This laser system is able to penetrate a water surface to provide bathymetric data of shallow water grounds. Here, a river section around a power plant in Rheinfelden, Germany, was acquired. The scan

was complemented by additional sonar measurements. We computed a digital terrain model (DGM) and a digital water model (DWM), resulting in a mesh consisting of 1880352 triangle elements with 944521 vertex nodes.

The ‘sea level’ of this data set has been set as 266.5m to correspond with the height of the water surface at power plant’s location. Figure 15 and Figure 16 provide views along the river using identical visualization parameters.

For comparison, contour lines were computed using a CPU marching squares algorithm, a simpler 2D variant of the 3D marching cubes [Lor87]. For each elevation meter a contour line was created in the Rheinfelden dataset, operating on the raw data representation as a point cloud (1.0m resolution). The computation time of the serial code was 12.5 seconds Intel-i7-2670QM@ 2.20GHz. Using parallelization via OpenMP, 8 threads on 8 CPUS reduced the computation time to 3.5seconds. In contrast, using an NVIDIA Quadro3000 on the same system, the triangular surface with procedural contour lines renders at 9.0 frames per second (fps) or 0.111 seconds. This results in a speed up of 32.1 in comparison to the OpenMP parallel CPU code for visual scene exploration or rendering of contour plots. If contour lines are required as explicit geometry for further processing, the shader-based procedural contour lines cannot replace the CPU algorithm.

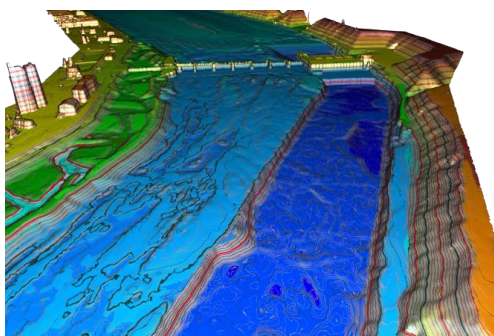


Figure 15 LIDAR-acquired bathymetry of a river power plant in Rheinfelden.

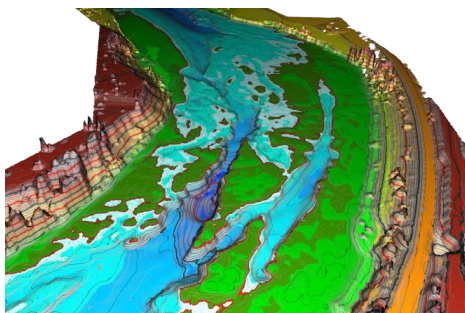


Figure 16 Bathymetry of a river bend at the Rheinfelden dataset, exposing some canyons.

Baltic LIDAR Terrain

Another LIDAR dataset was acquired by the same airborne laser scan system as used for the Rheinfelden area. The raw dataset provides a point cloud of 385 million elements. It was reduced to a triangular surface representing the ground surface (DGM) and a second triangular surface representing the water surface (DWM). The DGM has 1659958 vertices with 3228653 triangle elements and the DWM 2670748 vertices with 5271291 triangle elements. Rendering of the DGM results in a frame rate of 6.3 fps, rendering of both, DGM and DWM, in 2.9 fps, measured on an Intel-Xeon-X5650@2.67Ghz(x6) NVIDIA-Quadro5000 system.

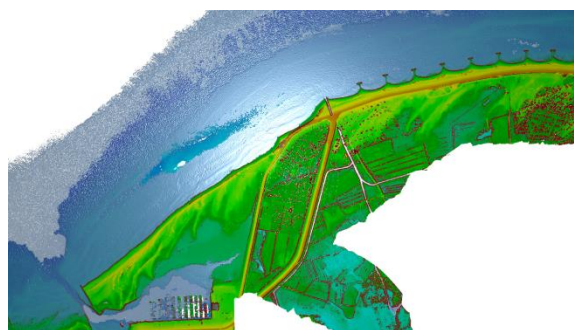


Figure 17: Variation in surface reflectivity. The coast line is illustrated by the border of the water surface extracted from the classified point cloud.

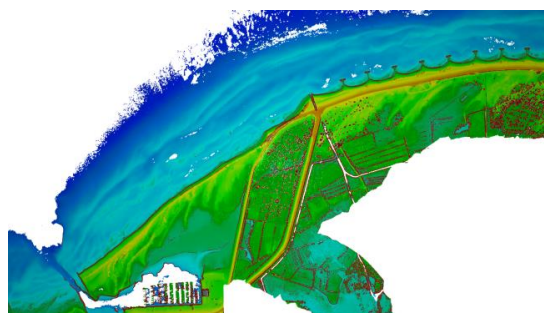


Figure 18: Highly sensitive hypsometric tinting illustrates structures on the ocean ground. The coast line matches the coast line of Figure 17.



Figure 19: Procedural contour lines emphasize the sand dunes on the ocean ground. Sand dunes have a length of about 50 to 800 meters.

Figure 17 demonstrates the effectiveness of varying the surface reflectivity to emphasize sea level. It shows how the coastline, illustrated by the shader (opaque surface), matches the coast line that was generated from laser echoes which had been classified as water surface (transparent surface). This is, for example, well-illustrated in the harbor region. Figure 18 provides an alternative view clearly depicting sub-surface structures on the ocean ground, such as sand dunes or the excavated harbor entrance. Figure 19 complements the colorization scheme with procedural contour lines. This scheme emphasizes the sand dune structures on the ocean ground.

4. PERFORMANCE

We found the impact of including procedural contour line in addition to usual color-coding of surfaces by height information to be negligible. For the Barataria Bay the rendering frame rate was found to be at most four percent slower when compared to rendering without contour lines. For the larger Rheinfelden dataset, the rendering frame rate of the grid was only 0.3 percent slower with contour lines than without. Table 1 summarizes our measurements. Since the rendering is view-dependent, the view point and direction influences the rendering time, so the compared views from the top, the front and the side. The values for the rendering frame rate are averaged from repeated execution of the actual OpenGL rendering call.

Name of the dataset	View	Frame rate with contour lines	Frame rate without contour lines	Relative increase of rendering frame rate
Barataria Bay	Top	341.764	357.782	4.4%
	Front	344.94	345.423	0.13%
	Side	346.86	359.066	3.3%
Rheinfelden	Top	17.139	17.1701	0.17%
	Front	16.983	17.0394	0.32%
	Side	17.149	17.1859	0.215%

Table 1: Rendering time performance of the grid with contour lines and without contour lines

To calculate the performance, the graphics processor used was a Quadro FX5600 with memory 1536 MB.

Also, eight CPU's with model name Intel® Xeon® CPU with memory 32GB was used.

These measurements merely refer to the performance of the shader. The triangular surfaces have been rendered as-is, without dynamic triangle refinement and level-of-detail. Those mechanisms, common from terrain rendering in particular when rendering height data given on a regular grid, are orthogonal to our shading technique and can be used to complement the overall performance, but are not part of our algorithm.

5. SUMMARY

In this article we presented a highly efficient method to render a height field given on a triangular mesh in a manner similar to well-known cartographic views, enhanced by essential or useful features such as line smoothing, gradient compensation, visibility fading, non-linear elevation mapping and shininess variation. The method can be implemented fairly simple on modern graphics cards to yield real-time renderings allowing precise quantitative assessment of data values. The effectiveness of this method has been demonstrated on a diverse set of geoscientific datasets from current research projects.

6. FUTURE WORK

It has been noted [Pat12] that the use of hypsometric tints may be misleading when representing terrains. E.g., low areas will always appear 'lush' even if they refer to a desert. Similarly open water above sea level will look as if it were on a land mass. This problem may be addressed by using cross-colorization, which is modifying the colors used in the LUT based on an additional scalar data field given on the terrain. Doing such requires a two-dimensional LUT and faces the challenge of non-repeating color entries to keep the rendering meaningful. In lieu or additionally, incorporating blending with RGB information from aerial photography may be useful as well. Also, alternative non-linear functions, in particular mappings based on histograms [Khu12], may be used to emphasize local details even better. The visual quality may well be enhanced by adding a geometry or tessellation shader to insert more triangles where needed, based on high order interpolation methods (i.e., dynamic level of detail). However, from a scientist's perspective to accurately identify their data elements, such an enhancement might not necessarily be desirable. Finally, the technique presented here is neither limited to elevation data nor to surfaces. All methods apply as well to a general scalar field replacing the elevation. Such may be given on a set of lines or non-planar surfaces as well, once the gradient of the scalar field and the base domain is known to allow for gradient compensation and visibility fading.

7. ACKNOWLEDGMENTS

This study was funded in part by BP through the grants to Louisiana State University and Coastal Waters Consortium. Special thanks to Frank Steinbacher for providing the LIDAR datasets. Thanks to Wolfgang Dobler for providing time measurements on the data set Rheinfelden. This work was supported by the Austrian Science Foundation FWF DK+ project Computational Interdisciplinary Modeling (W1227), and grant P19300. This research employed resources of the Center for Computation and Technology at Louisiana State University, which is supported by funding from the Louisiana legislatures Information Technology Initiative.

8. Additional Authors

Marcel Ritter, University of Innsbruck & Airborne Hydromapping OG, Technikerstrasse 13&21 A-6020, Innsbruck, Austria; Dubravko Justic, Department of Oceanography and Coastal Sciences, School of the Coast and Environment, 2221 Energy Coast and Environment, Louisiana State University, Baton Rouge, LA 70803, djusti1@lsu.edu; Lixia Wang, Department of Oceanography and Coastal Sciences, School of the Coast and Environment, 2223 Energy Coast and Environment, Louisiana State University, Baton Rouge, LA 70803, lxwang@lsu.edu

9. REFERENCES

- [Gul10] Gul, S.; Khan, M.F., "Automatic Extraction of Contour Lines from Topographic Maps," Digital Image Computing: Techniques and Applications (DICTA), 2010 International Conference on , vol., no., pp.593,598, 1-3 Dec. 2010
- [Du04] Du Jinyang; Zhang Yumei, "Automatic extraction of contour lines from scanned topographic map," Geoscience and Remote Sensing Symposium, 2004. IGARSS '04. Proceedings. 2004 IEEE International , vol.5, no., pp.2886,2888 vol.5, 20-24 Sept. 2004
- [Gro10] Grochow, K.; Stoermer, M.; Fogarty, J.; Lee, C.; Howe, B.; Lazowska, E., "COVE: A Visual Environment for Multidisciplinary Ocean Science Collaboration," e-Science (e-Science), 2010 IEEE Sixth International Conference on , vol., no., pp.269,276, 7-10 Dec. 2010
- [Ste10] F. Steinbacher, M. Pfennigbauer, A. Ulrich, and M. Aufleger, "Vermessung der Gewässersohle - aus der Luft - durch das Wasser," in Wasserbau in Bewegung ... von der Statik zur Dynamik. Beiträge zum 15. Gemeinschaftssymposium der Wasserbau Institute TU München, TU Graz und ETH Zürich, 2010.
- [Pat12] Patterson, T., & Jenny, B. (2012). The Development and Rationale of Cross-blended Hypsometric Tints. *Cartographic Perspectives*, 0(69), 31-46. Retrieved from <http://www.cartographicperspectives.org/index.php/journal/article/view/cp69-patterson-jenny>
- [Ino08] Inoue, M., Park, D., Justic, D., Wiseman, W. J., Jr. 2008. A High-Resolution Integrated Hydrology-Hydrodynamic Model of the Barataria Basin System. *Environmental Modelling and Software* 23: 1122-1132.
- [Das10] Das, A., Justic, D., Swenson, E. 2010. Modeling estuarine-shelf exchanges in a deltaic estuary: Implications for coastal carbon budgets and hypoxia. *Ecological Modelling* 221: 978-985.
- [Das11] Das, A., Justic, D., Swenson, E., Turner, R. E., Inoue, M., Park., D. 2011. Coastal land loss and hypoxia: The 'outwelling' hypothesis revisited. *Environmental Research Letters* 6: 025001 (9 pp); doi:10.1088/1748-9326/6/2025001.
- [Das12] Das, A., Justic, D., Inoue, M., Hoda, A., Huang, H., Park., D. 2012. Impact of Mississippi River diversions on salinity gradients in a deltaic Louisiana estuary: Ecological and management implications. *Estuarine, Coastal and Shelf Science* 111: 17-26.
- [Vin03] Leonardo da Vinci, A map of central Italy, 1503, Royal Collection by 1690; available from <http://www.royalcollection.org.uk/eGallery/object.asp?maker=12196&object=912277&row=436&detail=magnify>
- [Lor87] Lorensen, William and Harvey E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics (SIGGRAPH 87 Proceedings)* 21(4) July 1987, p. 163-170)
- [Khu12] S. Khurana, N. Brener, B. Karki, W. Benger, S. Roy, S. Acharya, M. Ritter, S. Iyengar, Multi Scale Color Coding of Fluid Flow Mixing Indicators along Integration Lines, WSCG2012, Plzen, 2012

BRDF Reconstruction Using Compressive Sensing

Nurcan Seylan

Computer Engineering
Yaşar University, Turkey

Ege Higher Vocational School
Ege University, Turkey

nurcan.seylan@ege.edu.tr

Serkan Ergun

International Computer Institute
Ege University, Turkey

serkan.ergun@ege.edu.tr

Aydın Öztürk

Department of Computer
Engineering
Izmir University, Turkey

aydin.ozturk@izmir.edu.tr



Figure 1: A sample scene has been rendered using alum-bronze and blue-metallic-paint materials from MERL database [9]. Left: Image based on the original data. Right: Image based on the reconstructed data using only 5% of the original (The Peak-to-Signal Ratio is 51.63 dB).

ABSTRACT

Compressive sensing is a technique for efficiently acquiring and reconstructing the data. This technique takes advantage of sparseness or compressibility of the data, allowing the entire measured data to be recovered from relatively few measurements. Considering the fact that the BRDF data often can be highly sparse, we propose to employ the compressive sensing technique for an efficient reconstruction. We demonstrate how to use compressive sensing technique to facilitate a fast procedure for reconstruction of large BRDF data. We have showed that the proposed technique can also be used for the data sets having some missing measurements. Using BRDF measurements of various isotropic materials, we obtained high quality images at very low sampling rates both for diffuse and glossy materials. Similar results also have been obtained for the specular materials at slightly higher sampling rates.

Keywords

BRDF reconstruction, compressive sensing.

1 INTRODUCTION

Real world materials display different reflection characteristics. Accurate representation of the distribution of light reflected from the surface of a material has long been studied in computer graphics. A class of functions called Bidirectional Reflectance Distribution Function (BRDF) defined in terms of incoming and outgoing light directions is commonly used to describe such reflectance properties.

Various models have been proposed for approximating the BRDF. It has been shown that some of these models meet the reciprocity and energy conserving principles.

However they generally fail to capture the reflectance properties of all material types. A natural approach to tackle this problem is to fit the underlying models to measured BRDF data. Since small numbers of parameters are involved in these models, only the corresponding estimates need to be stored for reconstruction of BRDF. Fitting can also be performed on data sets having some missing measurements. Nevertheless, such fitting procedure leads to some approximation errors for certain materials and its implementation is difficult in most cases because of its computational complexity [12].

A general and simple method for approximating the BRDF would be to use directly the measured BRDF data which is obtained on a regular grid using some version of gonioreflectometers. Then the intermediate BRDF values can be estimated by an interpolation. However, the BRDF data obtained in this way is generally noisy and contains some missing observations due to some difficulties in measuring BRDF around grazing angles [14].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

A major difficulty of using the BRDF data is its large size. Even if the raw data were correct and complete, its size would be prohibitively large for an efficient storage and for a rendering application. An alternative approach is to compress the measured BRDF data using some well-known compression techniques. These techniques are based on using basis functions (splines, spherical harmonics, wavelets, and Zernike polynomials), dimension reduction techniques (Principal Component Analysis, Independent Component Analysis and Cluster Analysis) and matrix factorization (Non-negative Matrix Factorization and Tensor Products). Empirical results have shown that these compression based techniques can provide an accurate and compact representation of BRDF data but do not offer an efficient importance sampling [8].

Generally, BRDF measuring systems suffer from occlusion problems because of using cameras, projectors, or even mirrors. In such cases, acquiring BRDFs over a full hemisphere may not always be possible. In some other cases measurements taken at certain angles may be prohibitively noisy and cannot be used for rendering. A possible approach for handling this problem is to ignore the missing or highly noisy measurements and fit an analytical model to the remaining part of the data [12]. Clearly the resulting fitting will not be adequate due to increased degree of the lack-of-fit of the model.

In this work we propose to employ an interesting technique namely the compressive sensing that can be used to reconstruct the missing BRDF measurements efficiently. It turns out that the proposed technique also provides an effective way of compressing the BRDF data.

The compressive sensing (which is also referred in the literature as compressed sensing or compressive sampling) has been evolving rapidly [3]. This technique takes advantage of the sparseness or compressibility of the data, allowing the entire measured data to be recovered from relatively few measurements using some optimization techniques. It has been shown that compressive sensing can also be used for data sets that contain some missing measurements [6].

We apply the compressive sensing approach on a large BRDF data for rendering applications. Based on the empirical results, we show that compressive sensing technique can be used effectively for image reconstruction. In Figure 1, we present rendered images based on measured BRDF data sets with 95% of its elements removed randomly and reconstructed on the right, and the original image on the left. This example illustrates the power of the proposed technique for reconstructing measured BRDF data using only a small portion of it. We also demonstrate the effectiveness of compressive

sensing technique for reconstruction of BRDF data having some missing or noisy measurements.

The paper is organized as follows: In Section 2 we explain briefly the compressive sensing technique. In Section 3 we present the problems encountered during BRDF data acquisition. In Section 4 we describe our reconstruction algorithm and in Section 5 we show experimental results. Section 6 is devoted to conclusions and discussions.

2 COMPRESSIVE SENSING

Compressive sensing technique, emerging over the past few years, has attracted considerable attention in digital signal processing. In this section we summarize compressive sensing technique for completeness. A good treatment of the topic may be found in [2, 5].

An n -dimensional signal is called sparse if it can be represented as a linear combination of smaller number of some basis vectors. The key idea behind compressive sensing technique is that sparse signals can be reconstructed perfectly in terms of smaller number of basis vectors.

Suppose that discrete time signals are represented by an $n \times 1$ column vector \mathbf{x} . Without loss of generality, higher dimensional data can also be represented by a vector by making an appropriate arrangement of the signal measurements. For example four dimensional BRDF data with a resolution $n = n_1 \times n_2 \times n_3 \times n_4$ where $n_i, (i = 1, 2, 3, 4)$ is the resolution of the i^{th} dimension, can be viewed as an $n \times 1$ vector. It is well known that a vector can be transformed into another $n \times 1$ vector \mathbf{s} , through an $n \times n$ orthogonal basis matrix Ψ as

$$\mathbf{x} = \Psi \mathbf{s} \quad (1)$$

Since Ψ is an orthogonal matrix, this equation can be solved for \mathbf{s} as $\mathbf{s} = \Psi' \mathbf{x}$ where Ψ' is the transpose of Ψ . If \mathbf{s} is known or can be estimated from the sample data then \mathbf{x} can be reconstructed easily from the above equation. This representation similar to that of principal components in the sense that the vectors \mathbf{x} and \mathbf{s} are the equivalent representations of the signals. In principal components representation, Ψ is determined adaptively from the sample data and the first $k \leq n$ nonzero entries of \mathbf{s} corresponding to significant row vectors in Ψ are used to recover \mathbf{x} .

Compressive sensing technique uses the sparsity property of the signals. If a signal is sparse then some of the entries of \mathbf{s} in Eq. (1) is expected to be zero leading to a representation with a reduced dimensionality. The underlying approach provides a non-adaptive technique where entries of the matrix is fixed and only a small

portion of the sample data is used for reconstruction of the vector \mathbf{x} .

Suppose that a signal is k -sparse that is only k entries of \mathbf{s} in Eq. (1) is nonzero. Let $\mathbf{y} = \boldsymbol{\varphi}\mathbf{x}$ where $\boldsymbol{\varphi}$ is an $m \times n$ sampling matrix ($m \leq n$) and \mathbf{y} is an $m \times 1$ vector. From Eq. (1) \mathbf{y} can be expressed as

$$\mathbf{y} = \boldsymbol{\varphi}\mathbf{x} = \boldsymbol{\varphi}\boldsymbol{\Psi}\mathbf{s} = \boldsymbol{\Theta}\mathbf{s} \quad (2)$$

where $\boldsymbol{\Theta}$ is an $m \times n$ transformation matrix, $\boldsymbol{\Psi}$ and \mathbf{s} are defined as in Eq. (1). Clearly, this system of m simultaneous linear equations with n unknowns cannot be solved for \mathbf{s} as the number of independent linear equations is much less than the number of unknowns. In a special case when signals are assumed to be k -sparse then a solution could be possible if the locations of the nonzero coefficients are known and $m \geq k$. A necessary and sufficient condition is that the transformation matrix should not change the lengths of the k -sparse vectors [1]. It has been shown that this condition is satisfied if the sampling matrix $\boldsymbol{\varphi}$ is chosen to be an identically and independently distributed gaussian matrix. An interesting result with this Gaussian matrix is that k -sparse signals of length n can be reconstructed using only $m \times 1$ vector \mathbf{y} where $m \geq ck \log(n/k) < n$ and c is a small constant random number generated from a Gaussian distribution.

The reconstruction algorithm for a k -sparse sample of size n should be able to determine the k nonzero and $(n - k)$ zero entries in $n \times 1$ vector \mathbf{s} . Finding the best combination out of m nonzero and $n - m$ zero combinations of the entries in \mathbf{s} is difficult. However an approximate solution can be obtained by minimizing the quantity

$$\xi(\mathbf{s}) = \sum_{i=1}^n |s_i| \quad (3)$$

with the constraint $\mathbf{y} = \boldsymbol{\Theta}\mathbf{s}$ where $\mathbf{s} = (s_1, s_2, \dots, s_n)$ is the sparse coefficient vector [1]. This process is known as ℓ_1 -norm optimization. An interesting result with ℓ_1 -norm optimization is that it tends to concentrate the energy of the signals onto a few nonzero entries of \mathbf{s} as opposed to the least squares which tends to spread the energy around. The reconstruction algorithm then consists of the following steps:

- i Determine an $m \times n$ sampling matrix $\boldsymbol{\varphi}$
- ii Obtain the $m \times 1$ measurement vector as $\mathbf{y} = \boldsymbol{\varphi}\mathbf{x}$
- iii Determine an $n \times n$ orthogonal basis matrix $\boldsymbol{\Psi}$
- iv Find the coefficient vector \mathbf{s} using ℓ_1 -minimization
- v Reconstruct \mathbf{x} using Eq. (1).

3 BRDF DATA WITH MISSING OR NOISY MEASUREMENTS

Commonly, BRDF measurements are obtained using a gonireflectometer, a computer controlled device which

typically has a photometer and a light source. Often the underlying system requires huge amount of measurements. For example, when an angular resolution of 1 degree is used, with a uniform sampling then the underlying system would require approximately one and a half million of measurements. It has been reported that measurements of reflectance at grazing angles are difficult to obtain accurately. For example, in evaluating several analytical BRDF models, Ngan et. al. [12] have ignored the data within an incoming and outgoing angles greater than 80 degrees considering that they are in general unreliable. In some other cases the optical elements of the system do not allow measurements at all at certain positions resulting considerable amount of missing data [11]. Experimental results have shown that approximately 60-65% of the measurements taken at the grazing angles and 10-15% of the measurements in the remaining region contain some reciprocity related errors [8]. On the other hand, Romerio et. al. [13] have mentioned about the existence of lens flare artifacts in BRDF measurements.

4 RECONSTRUCTION OF BRDF MEASUREMENTS

In this work, the compressive sensing technique is applied on isotropic BRDF data which is assumed to have some missing measurements. For this purpose the three dimensional data is divided into sub-sample blocks of size $15 \times 15 \times 15$. Random samples were generated from these sub-samples at a predefined sampling ratio. Finally, the resulting uniform random samples were used to reconstruct the underlying blocks employing the compressive sensing technique. An ℓ_1 -norm optimization algorithm proposed by van den Berg and Friedlander [16] was used for estimating the sparse vector \mathbf{s} as defined in Eq. (1).

As was explained in the preceding section, the compressive sensing technique requires using a sampling matrix $\boldsymbol{\varphi}$, and an orthogonal basis matrix which is incoherent with this sampling matrix. A number of sampling methods have been proposed for reconstructing signal data [3, 4]. Unfortunately, these methods cannot be applied on BRDF data directly unless an appropriate sampling strategy is used for obtaining the BRDF measurement. It is obvious from Eq. (2) that when the vector \mathbf{x} contains some missing data points, that is when some of the entries are missing, then the corresponding dot product between the vector \mathbf{x} and the rows of the Gaussian matrix $\boldsymbol{\varphi}$ cannot be determined. To overcome this difficulty, we proceed to use a different sampling procedure namely point sampling which is based on using a permutation matrix instead of a random Gaussian matrix. It is shown that the permutation matrices are coherent with the basis matrices which produce highly sparse data like the ones that are based

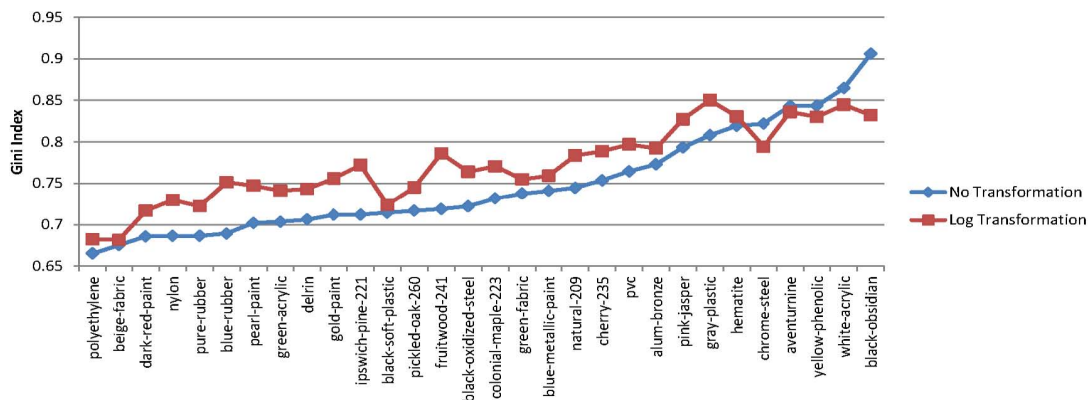


Figure 2: Computed Gini's indices of sparsity coefficients obtained in Fourier domain for BRDF measurements of 30 isotropic materials. Higher index values are found with log transformations.

on wavelets [15]. In our work we created an $n \times n$ basis matrix Ψ whose entries are obtained through Fourier transforms. It is reported that Fourier basis matrices often do not produce sparse coefficients in vector \mathbf{s} for real-world images [15]. However, our empirical results based on BRDF data have shown that highly sparse coefficients can be obtained with Fourier basis matrices when they are used with permutation matrices (point sampling). This property of using Fourier basis matrices is demonstrated in Figure 2. In this figure, Gini's indices are obtained and plotted for each material. Similar results based on log transformations of BRDF are also obtained and shown on the same figure. Higher values of Gini's index corresponds to higher sparsity of the measured data. It is seen that, the Gini's indices obtained for this case is found in the range (0.68, 0.85).

The permutation matrix which consists of zeros and ones are provided by generating these numbers using a simple random sampling technique without replacement. During the sampling process, if an entry of this matrix corresponding to a missing value in the vector \mathbf{x} is 1 then it is set to 0 and the next available position is set to 1.

It is assumed that the BRDF measurements must be positive [10]. However some data sets contain negative values as a result of certain sampling errors. We used log transform of the BRDF measurements to preserve

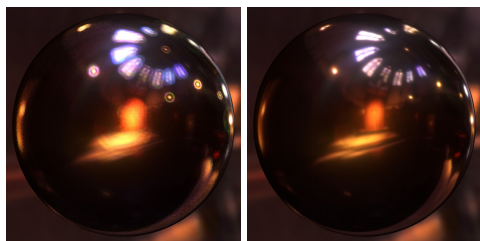


Figure 3: Left: Image based on BRDF measurements. The presence of lighting artifacts due to negative BRDF values. Right: Image obtained with log transformation.

the underlying property of BRDF. Our empirical results have shown that such transformation also increases the sparsity of the BRDF data. This situation is illustrated in Figure 3. It is seen in the figure that negative values cause some artifacts under illumination.

5 RESULTS

To demonstrate the efficiency of the compressive sensing approach, we considered a data set based on various isotropic materials acquired by Matusik et.al. [10] from MERL MIT database [9]. In this data set 1458000 measurements are provided for each material. We selected 30 isotropic materials to represent various diffuse and reflection properties. ℓ_1 -norm estimates of the coefficient vectors for each material were computed in Fourier domain. Gini's index [7] is used as a measure of sparsity and computed for each case.

It is seen in Figure 2 that in 25 materials out of 30, the sparsity indices based on log transformation are found to be higher than those based on the original data.

To investigate the effect of the sampling ratio on the visual quality of the reconstructed images, random samples with ratios 1%, 2.5%, 5%, 10%, 25%, 50% were generated from six different materials namely dark-red-paint, green-fabric, blue-metallic-paint, gold-paint, fruitwood-241, chrome-steel were chosen. These materials were chosen to reflect the diffuse, glossy and specular properties of reflection. Rendered spheres based on the original data is shown in the first row of the Figure 4 while the reconstructed images are presented in the following rows. The insets for each sphere represents the difference image between the corresponding reconstructed and the original images scaled by 8. The Peak-to-Signal(PSNR) values are also given for each reconstructed image.

It is interesting to see that images with a visually acceptable quality could be obtained by sampling only 1% of the measurements for diffuse and glossy mate-

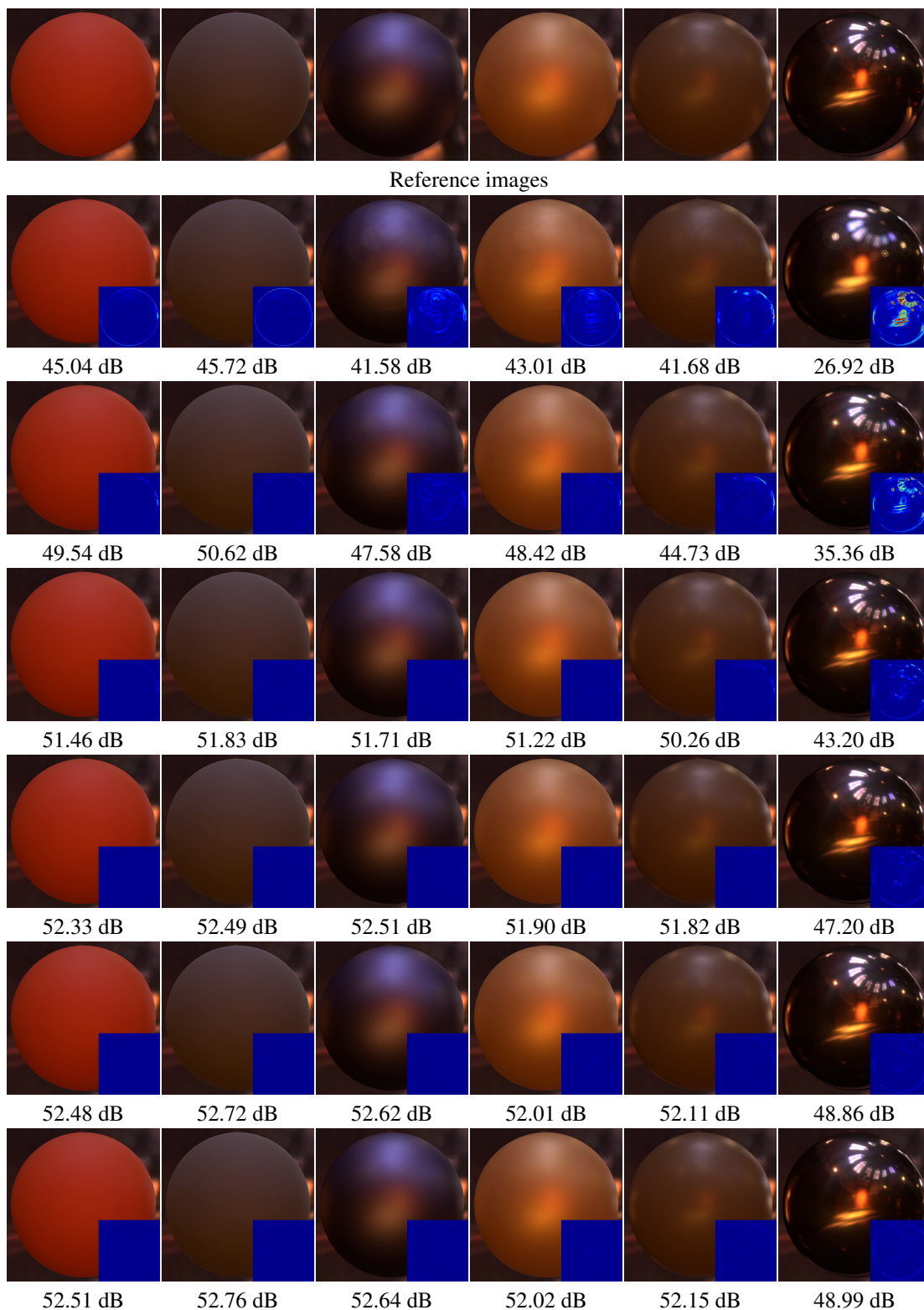


Figure 4: Rendered spheres under global illumination. First two columns: Diffuse materials (*dark-red-paint*, *green-fabric*), Third and fourth columns: Glossy materials (*blue-metallic-paint*, *gold-paint*), Last two columns: Specular materials (*fruitwood-241*, *chrome-steel*) [9]. First row: Original images. Second row through seventh row: Images obtained at 1, 2.5, 5, 10, 25, and 50 percent. Insets indicate the scaled differences between the given image and the corresponding original image.

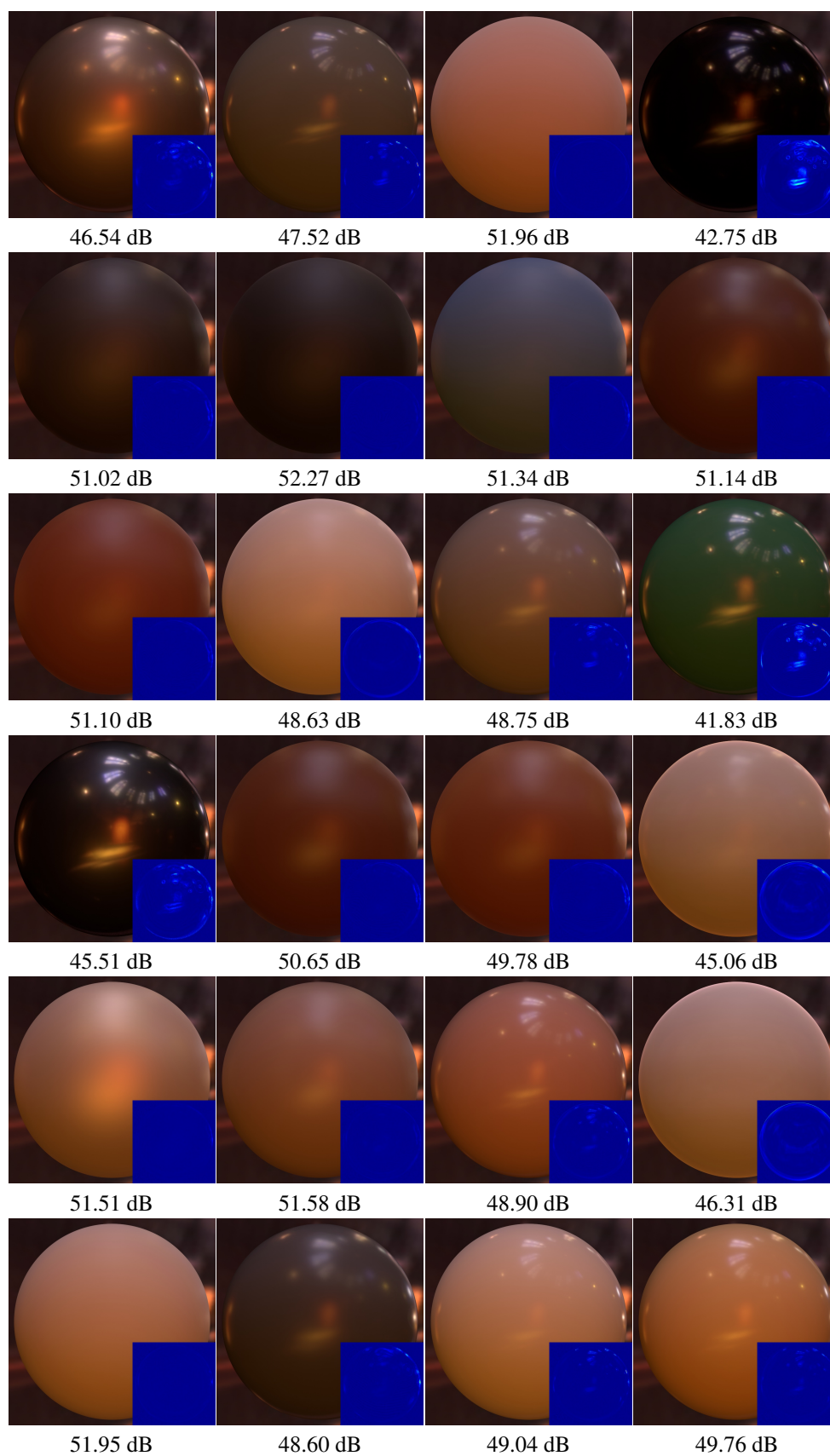


Figure 5: Reconstructed images using 5% of the BRDF measurements of randomly selected 24 isotropic materials from MERL data base. Insets indicate the differences between the given image and the corresponding original image. PSNR values are given for each material.

rials. In all cases except the first two cases for chrome-steel corresponding to 1% and 2.5% sampling ratios, the PSNR values are above 40 db. These results demonstrate the power of the compressive sensing approach when dealing with BRDF data having missing measurements. It can also be seen from Figure 5 that compressive sensing approach produces visually acceptable quality for all material types by sampling only 5% of the original data.

6 CONCLUSIONS AND DISCUSSIONS

In this work we analyzed the potential use of compressive sensing technique to facilitate a fast procedure for processing large BRDF data. In image reconstruction, compressive sensing can be more efficient than traditional sampling when data is sparse. Considering the fact that the BRDF data often can be highly sparse, it can be reconstructed efficiently using compressive sensing technique. We have demonstrated that the proposed technique can also be used for the data sets having some missing or unreliable measurements. Using BRDF measurements of various isotropic materials, we have shown that high quality images can be reconstructed at very low sampling ratios both for diffuse and glossy materials. Similar results also have been obtained for the specular materials at slightly higher sampling ratios.

It is well known that modeling and representation of anisotropic data is difficult. More data acquisition is needed for this case as compared with isotropic materials. We expect that the proposed approach can be extended to BRDF reconstruction for anisotropic materials.

7 REFERENCES

- [1] R.G. Baraniuk. Compressive sensing [lecture notes]. *Signal Processing Magazine, IEEE*, 24(4):118–121, 2007.
- [2] E.J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, 2006.
- [3] Emmanuel J. Candes, Justin K. Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006.
- [4] Thong T. Do, Lu Gan, Nam H. Nguyen, and Trac D. Tran. Fast and efficient compressive sensing using structurally random matrices. *IEEE Transactions on Signal Processing*, 60(1):139–154, 2012.
- [5] David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.
- [6] Jort Gemmeke and Bert Cranen. Missing data imputation using compressive sensing techniques for connected digit recognition. In *Proceedings of the 16th international conference on Digital Signal Processing, DSP'09*, pages 37–44, Piscataway, NJ, USA, 2009. IEEE Press.
- [7] Niall Hurley and Scott Rickard. Comparing measures of sparsity. *Machine Learning for Signal Processing, 2008. MLSP 2008. IEEE Workshop on*, pages 55–60, Oct. 2008.
- [8] Jason Lawrence, Szymon Rusinkiewicz, and Ravi Ramamoorthi. Efficient BRDF importance sampling using a factored representation. *ACM Trans. Graph.*, 23(3):496–505, 2004.
- [9] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. Merl BRDF database. <http://www.merl.com/brdf/>.
- [10] Wojciech Matusik, Hanspeter Pfister, Matthew Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Trans. Graph.*, 22(3):759–769, 2003.
- [11] Wojciech Matusik, Hanspeter Pfister, Matthew Brand, and Leonard McMillan. Efficient isotropic BRDF measurement. In *Proceedings of the 14th Eurographics workshop on Rendering, EGRW '03*, pages 241–247, Aire-la-Ville, Switzerland, 2003. Eurographics Association.
- [12] Addy Ngan, Frédo Durand, and Wojciech Matusik. Experimental analysis of brdf models. In *Proceedings of the Eurographics Symposium on Rendering*, pages 117–226. Eurographics Association, 2005.
- [13] Fabiano Romeiro, Yuriy Vasilyev, and Todd Zickler. Passive reflectometry. In David A. Forsyth, Philip H. S. Torr, and Andrew Zisserman, editors, *ECCV (4)*, volume 5305 of *Lecture Notes in Computer Science*, pages 859–872. Springer, 2008.
- [14] Szymon Rusinkiewicz. A survey of BRDF representation for computer graphics. *CS348c. Universidad de Stanford*, 1997.
- [15] Pradeep Sen and Soheil Darabi. Compressive rendering: A rendering application of compressed sensing. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):487–499, April 2011.
- [16] Ewout van den Berg and Michael P. Friedlander. Probing the Pareto frontier for basis pursuit solutions. *SIAM J. Sci. Comput.*, 31(2):890–912, November 2008.

Influence of Dynamic Wrinkles on the Perceived Realism of Real-Time Character Animation

Javier Alcon
 Universidad Rey Juan
 Carlos
 Modeling and Virtual
 Reality Group
 c/ Tulipàn, s/n
 28933, Mòstoles, Spain
 Javier.Alcon@gmail.com

David Travieso
 Universidad Autonoma
 de Madrid
 Perception and Action
 Research Group
 c/ Ivan Pavlov, 6
 28049, Madrid, Spain
 David.Travieso@uam.es

Caroline Larboulette
 University of South
 Brittany
 IRISA-UBS Lab
 Campus de Tohannic
 56000, Vannes, France
 Caroline.Larboulette@gmail.com

ABSTRACT

When creating a real-time character animation (e.g. for video games), fine grain details such as cloth wrinkles are very often omitted due to the limitation of available resources. However, we believe that they are crucial to increase realism, hence user immersion. In this paper, we present a perceptual study of dynamic wrinkles on clothes to show that they indeed have an important impact on realism. The models, animations and textures we used are *game* realistic and the wrinkling algorithm is fast enough to be used in a game. We have found that parameters influencing the perception of dynamic wrinkles include movement type and speed, viewing angle, and texture colors. Our results should be useful for animators or game designers to help them add wrinkles only where they are necessary, and activate them only when they have an impact on realism.

Keywords

Perceptual Study, Wrinkles, Character Animation

1 INTRODUCTION

According to game industry professionals [Buc05a], the overall perception of a virtual character can be split into three main components: the rendering (*how it looks*), the animation (*how it moves*), and the artificial intelligence (*how it behaves*). For a consistent perception of a virtual environment, it is essential that these three aspects be in harmony.

In real-time animations however, secondary animations such as wrinkles of skin or cloth, soft tissues dynamics or muscles are generally omitted because their impact on realism is unsure for an increased cost in computation time and memory consumption. This yields to a large difference in quality between the rendering and the animation.

Low cost techniques such as [Lar04a, Lar05a, Dec06a, Ram09a] have been developed to add details to character animations but no study so far has shown their effectiveness. Alternatively, a number of physically based

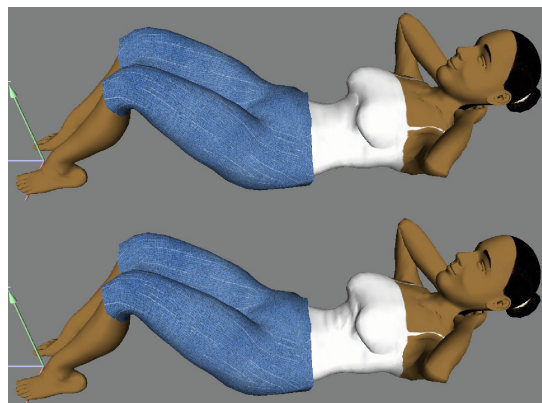


Figure 1: Two frames of two different videos included in our experiment: top- without dynamic wrinkles; bottom- with dynamic wrinkles in the abdominal region. Other parameters such as movement, viewing angle and textures are identical.

techniques exist that produce more accurate dynamic deformations [Jam02a, Cap02b], wrinkles [Wan10a] or muscles [Lee09a]. However, *physically correct* does not mean *perceived as realistic*, simply because humans have wrong beliefs and expectations that are often not physically correct [Bar96a]. The opposite is also true, because there are many different but similar motions that can be generated from roughly the same initial con-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ditions with unknown variables (for example, an unknown wind gust changing the trajectory of a baseball), humans will perceive all of them as *visually plausible* although a physically based simulation would always produce the exact same movement. It is thus necessary to study the impact of secondary animations on realism by including human perception into the design of efficient algorithms.

In this paper, we propose a study of dynamic wrinkles on clothes of characters in a game environment. We first study the impact of dynamic wrinkles on realism for different movements and viewing angles (section 4). We then study the influence of texture colors on the perception of dynamic wrinkles and overall realism (section 5). This is done by showing the same videos with or without dynamic wrinkles in all the possible configurations to participants (see figure 1). We finally expose a summary of our findings before proposing future directions of study (section 6).

2 RELATED WORK

Perceptual Studies

Recently, researchers have started to investigate the role of perception to reduce rendering times where it was not necessary. To mention only a few, [Cat03a] showed the use of task maps in a task oriented environment to help select the areas of interest that should be rendered with more accuracy. [Sun04a] studied to what level viewers notice degradations in image quality when the quality of the rendered image decreases. It has been then discussed [Osu04a] how human perception could be taken into account to animate deformable objects and humans in virtual environments.

Early work [Oes00a] on the perceptual study of character animation in virtual environments showed that even if observers are often unaware of specific details, they will still perceive an animation with a different level of realism. We observed the same in our study as participants graded better the videos with dynamic wrinkles although most of them were not aware of what was different from one animation to another.

More directly related to our work, [Mcd07a] studied the minimal pose update rate necessary to achieve smooth perception of movement of characters in a virtual environment. They detect that cycle rate, linear velocity (which is how fast a character moves across the screen) and movement complexity influence the perceived smoothness of a movement. In addition, the *contrast sensitivity function* defines a range of perception of black/white patterns that depends on the contrast and spacial frequency of the patterns. There is an optimal size of such patterns that makes their visual perception higher [Lue01a].

In our study, to avoid any uncontrolled effect due to the change of the contrast and spatial frequency, we kept

both illumination and distance of the character to the viewer constant. In addition, the camera followed the character that remained in the center of the screen, so we had no linear velocity. However, we studied the effect of dynamic wrinkles on animations with different movement complexities.

[Mcd08a] showed that it is possible to generate a crowd composed of different characters using a single 3D model by changing the textures used for the clothes, hair and skin. While it is efficient as it reduces the ability of people to spot clones, we have been wondering how that would affect the realism and the perception of wrinkles in such an environment. We have thus studied how the perception of wrinkles is affected when the texture of clothes and the skin color of a character are changed.

In [Cou09a], the impact of facial expressive wrinkles is studied in relationship with emotions. We study the impact of wrinkles of clothes on realism.

Dynamic Wrinkles

In the past ten years, a few algorithms have been proposed to add wrinkles to a character animation. They range from physically based [Kan02a, Wan10a] to geometrically based [Lar04a, Dec06a] techniques. Other methods to generate wrinkles include bump-mapping techniques [Ban02a, Wu96a]. As we show, bump-mapping techniques fail to accurately represent wrinkles when viewed from the side as it corresponds to both, the angle where the wrinkles increase the most the realism, and the angle when the mesh appears flat.

Physically based techniques are more accurate but generally require more resources than geometrically based techniques, which makes them difficult to use in video games. While we keep the study of visually plausible versus physically accurate for a later study, we have decided to use the technique of [Lar04a] because the wrinkles generated are fast and simple, and in harmony with the type of animation and texture we have used.

3 FRAMEWORK

We have conducted 2 experiments for a total of 4 different analyses in order to study the influence of dynamic wrinkles on character animation in a video game environment (one character in real-time). To this end, we have used a simple real-time algorithm to add wrinkles [Lar04a] to an animation we have generated under Autodesk Maya [May13a]. The wrinkles are manually defined by a wrinkling curve that serves as the profile of the wrinkles. The wrinkles form perpendicularly to the profile and are attenuated towards the borders. During animation, the length of the wrinkling curve is kept constant by forming waves. The mesh vertices are then displaced according to this profile.

To be sure no bias was introduced due to contrast, the background color of each animation was automatically computed as a grey color, the brightness of which was computed as the average brightness of the textures. We also kept constant the scene lighting by using the default ambient OpenGL light that is non-directional.

In order to keep the spatial frequency of the wrinkles constant, we kept the distance of the character to the camera constant. In addition, we used a chin-rest as depicted in figure 2 so that the participants also remained at a constant distance to the screen, and this distance was the same for all participants.

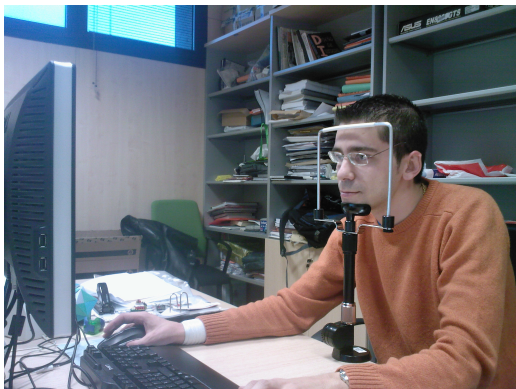


Figure 2: To fix the distance of the user to the screen, we used a chin-rest for all of our experiments.

Finally, we used the same hardware and software during each experiment and the conditions of luminosity in the room were fixed (closed curtains, constant illumination from the ceiling).

We have studied the perception of wrinkles according to the following parameters: type of movement, type of skinning, viewing angle and texture color (see figures 1, 3 and 8). The videos have been generated at 24 *fps* with a *MPEG-2 Blu-Ray* format to obtain high quality.

30 participants took part in each of the experiments, 20 males and 10 females in the first one, 14 males and 16 females in the second one, aged from 18 to 58. All of them had a normal vision. They came from various backgrounds and were more or less familiar with Computer Graphics and video games. They were not told what the experiment was about and received a chocolate bar for participating. Each experiment took about 5 to 10 minutes, depending on the participant.

In the *abdominals* animation, wrinkles were added on the character's belly (given that the rest of the character was static, no extra wrinkles were needed). In the *walk cycle* and *kick boxing* animations, the wrinkles were added on the pants, in the knee region, both in the front and in the back. The wrinkles parameters were the same for both animations.

The video sequence was randomly chosen for each participant to avoid any learning effect. Before starting the

video sequence, participants were explained the task on screen and asked a few questions of sex, age and background. They were asked to grade each video from 1 to 7 from *not realistic* to *highly realistic*. The participant was responsible for starting each video when he or she was sure to be ready to watch it.

For the statistical analyses, we used a repeated measures ANalysis Of VAriance (ANOVA) with three intra-participants factors, which means that each participant who took the test saw all of the videos (see Annex for details).

4 MOVEMENT AND VIEWING ANGLE EXPERIMENT

In the first experiment, the hypotheses were the following:

- dynamic wrinkles enhance the perception of realism of an animation even if those wrinkles are barely visible;
- this enhancement depends on the type of movement (fast movements versus slow movements);
- the perception of wrinkles depends on the viewing angle.

4.1 Experimental Setup

The variables of the experiment were:

- the type of movement:
 - *abdominals*, slow movement;
 - *walk cycle*, medium speed movement;
 - *kick boxing*, fast movement.
- the type of skinning (see figure 1):
 - without dynamic wrinkles;
 - with dynamic wrinkles.
- the position of the camera in relation to the character (see figure 3):
 - front (twice 0 degree);
 - diagonal (45 and -45 degrees);
 - side (90 and -90 degrees).

Thirty participants took part in the experiment, 20 males and 10 females. The experiment trials were run on a standard Dell PC under Windows Vista with an LCD monitor (see figure 2). Input from the user was recorded on the same computer, the videos were graded after each viewing.

We have shown 36 videos in total: 3 movements * 2 types of skinning * 6 different viewing angles. All of the videos lasted 5 seconds.

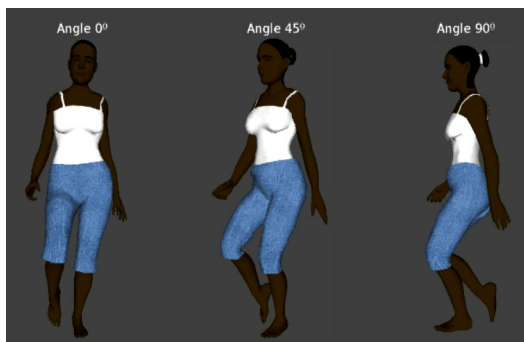


Figure 3: Three frames of three different videos included in our experiment with a viewing angle varying from 0 to 90 degrees.

4.2 Movement Analysis

A repeated measures ANOVA was performed on the perceived realism ratings, the within-subjects factors being the type of movement, the type of skinning and the position of the camera. There were three main effects: in the type of movement ($F_{(2,118)} = 7.17$, $p < 0.001$), in the type of skinning ($F_{(1,59)} = 5.617$, $p < 0.05$) and in the viewing angle ($F_{(2,118)} = 1.97$, $p < 0.01$). In addition, there was an interaction effect between the type of movement and the type of skinning ($F_{(2,118)} = 4.16$, $p < 0.018$). No more significant effects were found.

Main Effects

The main effect in the type of movement means that the type of movement influences the perception of realism, the *abdominals* animation being perceived as the most realistic one, followed by the *walk cycle* and the *kick boxing* animation. This can be seen on figure 4.

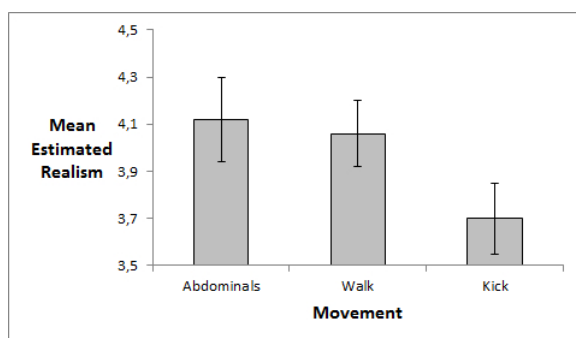


Figure 4: Mean estimated realism and SEM as a function of the type of movement (x-axis). Error bars show the standard error of the mean.

The main effect in the type of skinning shows that animations with dynamic wrinkles are much more realistic than animations without those wrinkles. This can be seen on figure 5.

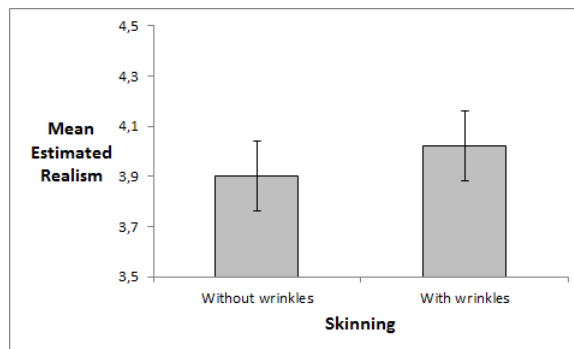


Figure 5: Mean estimated realism and SEM as a function of the type of skinning (x-axis). Error bars show the standard error of the mean.

Interactions

In addition, we detected an interaction between the type of movement and the type of skinning. The *abdominals* animation with dynamic wrinkles was perceived as the most realistic one, followed by the *walk cycle* with dynamic wrinkles, the *walk cycle* without the dynamic wrinkles, the *abdominals* without the dynamic wrinkles and the *kick boxing* animations, with and without wrinkles, with very little difference between both (see figure 6).

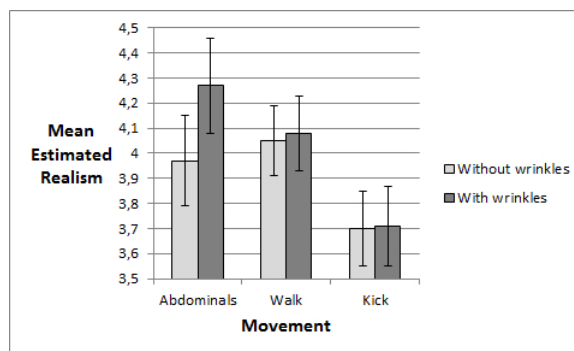


Figure 6: Mean estimated realism and SEM as a function of the type of movement (x-axis) and the type of skinning. Error bars show the standard error of the mean.

The results concerning the *abdominals* animation show that on a very simple and slow animation, the dynamic wrinkles are much more important for realism, as the difference between both bars is the biggest. Dynamic wrinkles are less important on fast movements as we can see for the *kick boxing* animation, probably because it is more difficult to perceive the wrinkles.

Moreover, the bars for the animations with dynamic wrinkles are always higher than the ones for the animations without wrinkles, which means that whatever the animation, dynamic wrinkles always increase realism, although this is better appreciated for slow movements.

Another interesting observation is that most people who took the test were not able to tell afterwards what we had tested. This means that the realism is increased, but people cannot tell why or what is different.

4.3 Viewpoint Analysis

The position of the character with respect to the camera had an effect on the perceived realism. As we can see on figure 7, when the character was seen from the side, it was perceived as more realistic than when seen from the front.

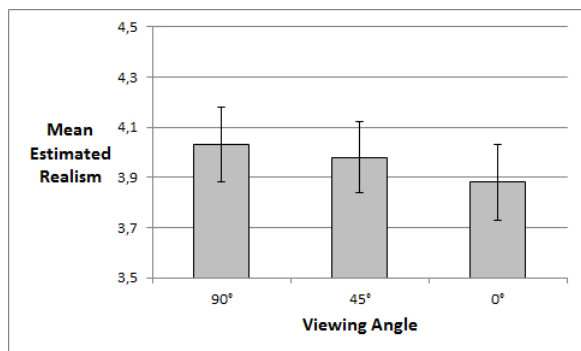


Figure 7: Mean estimated realism and SEM as a function of the viewing angle (x -axis). Error bars show the standard error of the mean.

More surprisingly however, we did not find any interaction between the viewing angle and the presence/absence of wrinkles. As the viewing angle is not constant in the *kick boxing* animation due to the fact that the character rotates, we first eliminated the corresponding video samples from the test to make a second analysis. We found an interaction between the movement type and the skinning type ($F_{(1,59)} = 5.514$, $p < 0.05$) as in our first analysis but still no interaction between the skinning type and the viewing angle.

We then also removed the *walk cycle* animation samples for two reasons. Firstly, there are wrinkles in the front and in the back of the pants, meaning that when the angle is of 45 degrees with the wrinkles in the front, it is 135 degrees with the ones in the back. Secondly, the wrinkles being on a non-uniform texture might create additional uncontrolled effects.

We thus did an analysis on the *abdominals* video samples only but we did not detect any effect ($F_{(2,118)} = 0.37$, $p = 0.0691$). As we will see in the second experiment, this was probably due to the fact that because we removed many samples, the analysis was done on only 12 ratings which is too little to obtain significant results.

5 CONTRAST EXPERIMENT

In the second experiment, we wanted to study the effect of texture colors on the perception of realism. As

the *abdominals* animation was the one where dynamic wrinkles had the highest effect on realism and the only one where the angle between the wrinkles and the viewing angle was meaningful, we used this animation for the experiment.

Our hypothesis was that wrinkles on the white top would be more visible than wrinkles on the black top, simply because the contrast would be higher. We verified this hypothesis and also discovered that the overall contrast with the skin of the character also influences the perception of realism of the animation.

As we were studying contrast, to avoid any effect due to the background color, its color was set to grey, the brightness of which was computed as the average brightness of the textures (see figure 8).

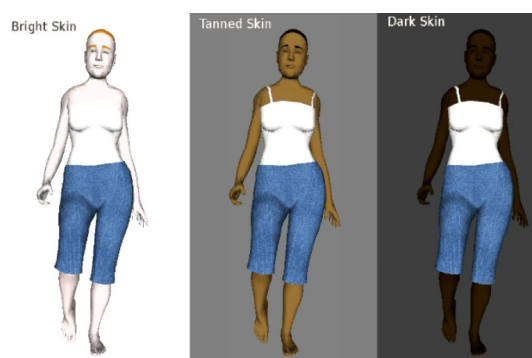


Figure 8: The same character with three different skin colors. The background color is a grey, the brightness of which is computed as the average brightness of the textures.

5.1 Experimental Setup

The variables of the experiment were:

- the type of texture:
 - bright skin, white top;
 - bright skin, black top;
 - tanned skin, white top;
 - tanned skin, black top;
 - dark skin, white top;
 - dark skin, black top.
- the type of skinning:
 - without dynamic wrinkles;
 - with dynamic wrinkles.
- the position of the camera in relation to the character:
 - front (twice 0 degree);
 - diagonal (45 and -45 degrees);

- side (90 and -90 degrees).

Thirty participants took part in the experiment, 14 males and 16 females. The experiment trials were run on a DELL XPS 1530 laptop PC, using the same conditions of room illumination and distance to the screen for all participants. Input from the user was recorded on the same computer, the videos were graded after each viewing.

In this experiment, we have shown a total of 72 videos: 6 different textures * 2 types of skinning * 6 different angles. All of the videos had a duration of 5 seconds.

5.2 Texture and Angle Analysis

Main Effects

A repeated measures ANOVA was performed on the perceived realism ratings, the within-subjects factors being the type of skinning, the texture and the position of the camera. There were main effects in the type of skinning ($F_{(1,59)} = 18.43, p < 0.001$), the viewing angle ($F_{(2,118)} = 21.88, p < 0.001$) and the texture ($F_{(5,295)} = 11.02, p < 0.001$). This analysis confirms the results of the first experiment regarding the influence of the presence of dynamic wrinkles and the viewing angle. It also shows that the texture influences the perception of realism as characters with white tops are rated more realistic than the ones with black tops (see figure 9).

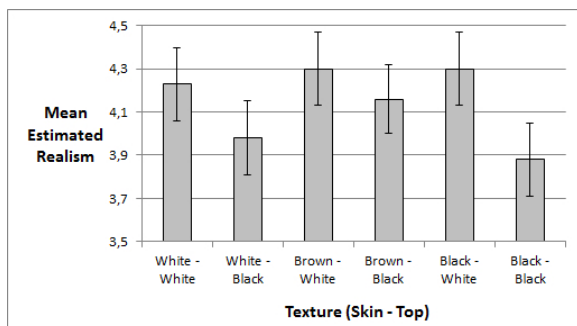


Figure 9: Mean estimated realism and SEM as a function of the texture (x -axis). Error bars show the standard error of the mean.

Interactions

We observed an interaction effect between the texture and the type of skinning ($F_{(5,295)} = 12.105, p < 0.001$) indicating that on the white top, the presence or absence of dynamic wrinkles makes a huge difference while on the black top, there is no significant difference (see figure 10).

We detected an interaction effect between the texture and the viewing angle ($F_{(10,590)} = 2.381, p < 0.01$).

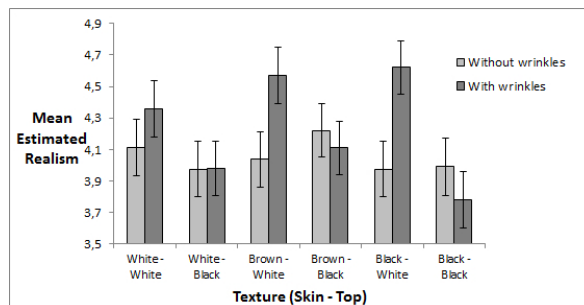


Figure 10: Mean estimated realism and SEM as a function of texture (x -axis) and skinning type. Error bars show the standard error of the mean.

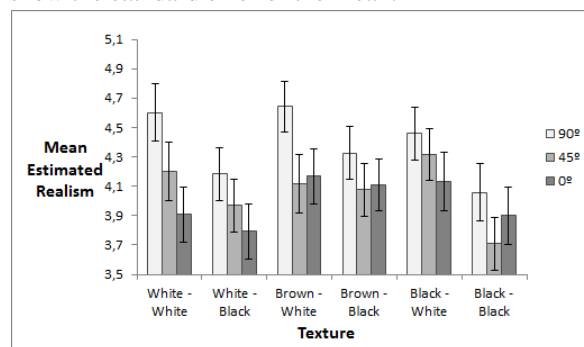


Figure 11: Mean estimated realism and SEM as a function of texture (x -axis) and viewing angle. Error bars show the standard error of the mean.

The effect is stronger for brighter tops and decreases for darker ones (see figure 11).

Finally, we also found an interaction effect between the type of skinning and the viewing angle ($F_{(2,118)} = 6.174, p < 0.005$). When the angle between the camera and the wrinkles profile is of 90 degrees (side view), the difference in perception of realism between the presence or absence of dynamic wrinkles is the highest, followed by an angle of 45 degrees and 0 degrees (front view). In the last case, there is almost no difference whether there are dynamic wrinkles or not. The same result was observed in our second analysis (see figure 16).

Those results confirm our hypothesis. Wrinkles increase realism more onto white tops. On the black tops, there is much less contrast possible, so the wrinkles cannot be properly perceived which makes the animations less realistic. In addition, when wrinkles are present, the realism increases when the viewing angle increases.

5.3 Skin Color Analysis

We have done a second analysis on the same data by removing the black tops and keeping only the white tops for the three different skin colors, the presence or absence of wrinkles and the six angles. This corresponds

to 36 videos in total: 3 different textures * 2 types of skinning * 6 different angles.

The aim of this analysis was to detect if the texture influences the perception of realism even if the wrinkles are not on the varying texture. That is why we kept the white tops only (i.e. when the wrinkles are more visible), but we used 3 different colors of skin. Our initial hypothesis would be that it would not influence the perception of realism.

Main Effects

As we predicted, the texture itself did not influence the realism of the animation which means that in the previous analysis, the difference in perceived realism was due to the top color, not to the skin color. We found main effects in the type of skinning ($F_{(1,59)} = 36.079, p < 0.001$, figure 12) and in the viewing angle ($F_{(2,118)} = 17.859, p < 0.001$, figure 13).

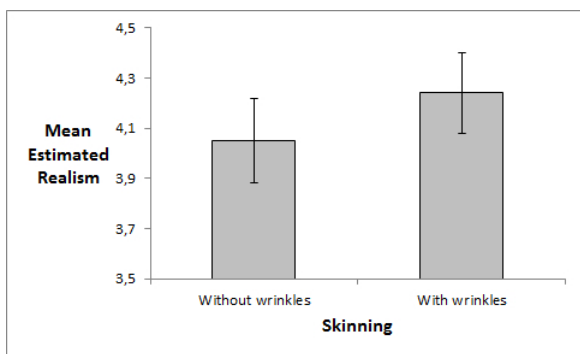


Figure 12: Mean estimated realism and SEM as a function of the type of skinning (x-axis). Error bars show the standard error of the mean.

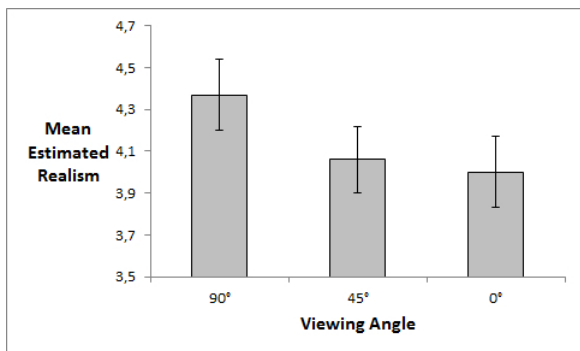


Figure 13: Mean estimated realism and SEM as a function of the viewing angle (x-axis). Error bars show the standard error of the mean.

Interactions

We found an interaction effect between the skin color and the type of skinning ($F_{(2,118)} = 5.547, p < 0.01$).

The darker the skin, the higher the contrast with the white top, the better perception of the presence of wrinkles. We can thus conclude that when dynamic wrinkles are present and when the contrast between the skin and the cloth increases, the animation is perceived as more realistic (see figure 14).

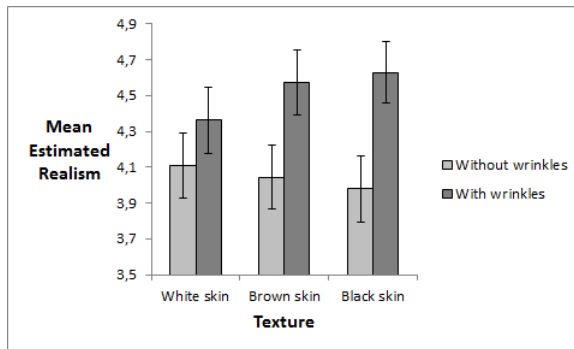


Figure 14: Mean estimated realism and SEM as a function of skin color (x-axis) and skinning type. Error bars show the standard error of the mean.

There was also an interaction effect between the skin color and the viewing angle as in our first analysis ($F_{(4,236)} = 2.442, p < 0.05$, figure 15).

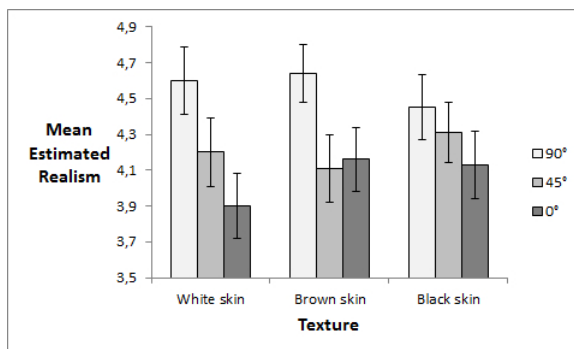


Figure 15: Mean estimated realism and SEM as a function of skin color (x-axis) and viewing angle. Error bars show the standard error of the mean.

Finally, we also found an interaction effect between the type of skinning and the viewing angle ($F_{(2,118)} = 16.17, p < 0.001$), which confirms that the second analysis of our first experiment done with only a few samples of the abdominals animation was invalid due to the small amount of data analyzed.

On figure 16, we can clearly see that when there is no dynamic wrinkles, the perception of realism is low and similar for all angles whereas when dynamic wrinkles are present, the realism increases when the viewing angle increases towards 90 degrees.

However, it would be interesting to create another experiment with different slow movements of a non-rotating character to reinforce our result.

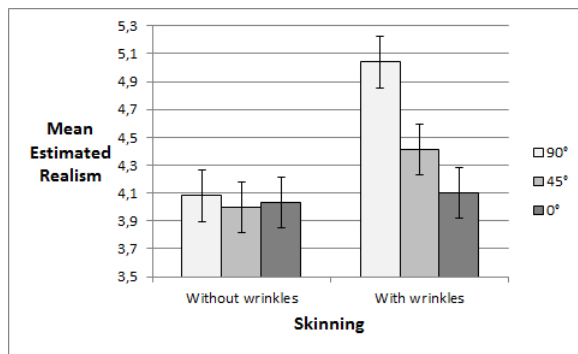


Figure 16: Mean estimated realism and SEM as a function of skinning type (x -axis) and viewing angle. Error bars show the standard error of the mean.

6 CONCLUSION AND FUTURE WORK

We have presented a perceptual analysis of the influence of dynamic wrinkles on the realism of a real-time character animation. We have shown that the presence of wrinkles always increases the realism of an animation, even if this gain is much lower for fast movements. As a fast movement usually necessitates more keyframes, a further study could be to determine what the minimal number of keyframes per second is so that dynamic wrinkles cannot be perceived anymore.

In addition, we have found that the viewing angle also influences the perception of realism, especially when wrinkles are viewed from the side. This means that bump mapping techniques [Wu96a, Ban02a] actually fail in simulating dynamic wrinkles because they are only accurate in the front view, i.e. when they are not needed, especially on dark surfaces. Wrinkles increase the realism a lot when seen from the side, something a bump mapping technique cannot achieve as the silhouette then appears flat.

As we initially hypothesized, dynamic wrinkles do not increase the realism as much on dark surfaces as it does on bright surfaces. From the front view, they were almost not detected. This is due to the contrast that is much lower (close to zero). As contrast is the only hint a viewer has in the front view, it is thus useless to create wrinkles on a black surface when only the default OpenGL ambient light is used for rendering.

A further study with more efficient lighting models would be needed to refine that result. More surprisingly, we have found that the overall contrast of the surface with wrinkles and the surrounding surfaces (skin color) also influences the perception of realism, the more contrast, the more realistic the animation.

One more useful parameter we did not include in this study is the distance parameter. It is obvious that from a certain distance, wrinkles won't be perceivable anymore. We would like to perform an additional experi-

ment to determine a threshold (ratio wrinkle size / distance) from which it is not necessary to include dynamic wrinkles as those won't be perceived anyhow.

Finally, we would like to study the effect of wrinkles generated with different algorithms. In this study, we have used video game technology for the character with a simple real-time technique to create the wrinkles. We would like to study, on the same character, the influence of dynamic wrinkles on realism, whether the algorithm is simplistic or more physically correct like in more recent work such as [Wan10a].

7 ACKNOWLEDGMENTS

We would like to thank all participants for their irreplaceable help. This research has been partially supported by project FFI2009-13416-C02-02 of the Spanish Ministry of Science and Innovation.

8 REFERENCES

- [Bar96a] Barzel R., Hughes J. F., Wood D. N. Plausible motion simulation for computer graphics animation. In *Computer Animation and Simulation'96 (Proceedings of the Eurographics Workshop)* (1996), pp. 184–197.
- [Ban02a] Bando Y., Kuratate T., Nishita T. A simple method for modeling wrinkles on human skin. In *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2002), IEEE Computer Society, p. 166.
- [Buc05a] Buchanan J. Invited presentation. ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games, Washington, USA.
- [Cat03a] Cater K., Chalmers A., Ward G. Detail to attention: Exploiting visual tasks for selective rendering. In *Eurographics Symposium on Rendering 2003* (June 2003), ACM, pp. 270–280.
- [Cap02b] Capell S., Green S., Curless B., Duchamp T., Popovic Z. Interactive skeleton-driven dynamic deformations. In *Proceedings of SIGGRAPH'02, ACM Transactions on Graphics* (July 2002), vol. 21, pp. 586–593.
- [Cou09a] Matthieu Courgeon, Stephanie Buisine, Jean-Claude Martin. Impact of expressive wrinkles on perception of a virtual character's facial expressions of emotions. *Intelligent Virtual Agents, Lecture Notes in Computer Science Volume 5773*, 2009, pp 201-214
- [Dec06a] Decaudin P., Julius D., Wither J., Boissieux L., Sheffer A., Cani M.-P. Virtual garments: A fully geometric approach for clothing design. *Computer Graphics Forum (Eurographics'06 proc.)* 25, 3 (sep 2006).

- [Gra08a] Gravetter F. J., Wallnau L. B. *Statistics for the Behavioral Sciences*, 8th ed. Wadsworth Publishing, dec 2008.
- [Jam02a] James D. L., Pai D. K. Dyr: Dynamic response textures for real time deformation simulation with graphics hardware. In *Proceedings of SIGGRAPH'02, ACM Transactions on Graphics* (San Antonio, TX, July 2002), vol. 21, pp. 582–585.
- [Kan02a] Kang Y.-M., Cho H.-G. Bilayered approximate integration for rapid and plausible animation of virtual cloth with realistic wrinkles. In *CA '02: Proceedings of the Computer Animation* (Washington, DC, USA, 2002), IEEE Computer Society, p. 203.
- [Lar04a] Larboulette C., Cani M.-P. Real-time dynamic wrinkles. In *Computer Graphics International* (2004), IEEE Computer Society Press, pages 522–525.
- [Lar05a] Larboulette C., Cani M.-P., Arnaldi B. Dynamic skinning: Adding real-time dynamic effects to an existing character animation. In *Spring Conference on Computer Graphics (SCCG)* (Budmerice Castle - Slovak Republic, May 2005), Juttler B., (Ed.), In cooperation with ACM SIGGRAPH and Eurographics, ACM Press.
- [Lee09a] S.-H. Lee, E. Sifakis, and D. Terzopoulos. Comprehensive Biomechanical Modeling and Simulation of the Upper Body. *ACM Transactions on Graphics*, 28(4):99:1–99:17, 2009.
- [Lue01a] Luebke D. P., Hallen B. Perceptually-driven simplification for interactive rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques* (London, UK, 2001), Springer-Verlag, pp. 223–234.
- [May13a] Maya Autodesk, 2013.
- [Mcd07a] McDonnell R., Newell F., O'Sullivan C. Smooth movers: perceptually guided human motion simulation. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, pp. 259–269.
- [Mcd08a] McDonnell R., Larkin M., Dobbyn S., Collins S., O'Sullivan C. Clone attack! perception of crowd variety. In *Proceedings of ACM SIGGRAPH 2008*, ACM Press, pp. 1–8.
- [Oes00a] Oesker M., Hecht H., Jung B. Psychological evidence for unconscious processing of detail in real-time animation of multiple characters. *Journal of Visualization and Computer Animation* 11, 2 (2000), 105–112.
- [Osu04a] O'Sullivan C., Howlett S., Morvan Y., McDonnell R., O'Connor K. Perceptually adaptive graphics. In *Proceedings of Eurographics 2004, State of the Art Reports*, Schlick C., Purgathofer W., (Eds.), INRIA and the Eurographics Association, pp. 141–164.
- [Ram09a] Ramos J., Larboulette C. Real-time anatomically based character animation. In *Proc. of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, Posters and Demos* (2009).
- [Sun04a] Sundstedt V., Chalmers A., Cater K., Debatista K. Top-down visual attention for efficient rendering of task related scenes. In *VMV 2004 - Vision, Modelling and Visualization* (November 2004), Stanford.
- [Wan10a] Wang H., Hecht F., Ramamoorthi R., O'Brien J. Example-based wrinkle synthesis for clothing animation. In *Proceedings of ACM SIGGRAPH 2010* (Los Angeles, CA, USA, 2010), ACM.
- [Wu96a] Wu Y., Kalra P., Thalmann N. M. Simulation of static and dynamic wrinkles of skin. In *CA '96: Proceedings of the Computer Animation* (Washington, DC, USA, 1996), IEEE Computer Society, p. 90.

ANNEX

ANOVA stands for ANalysis Of VAriance. The analysis is done using a dependent variable (the level of realism from 1 to 7 in our experiments) and a number of independent variables such as movement type, skinning type, viewing angle and texture in our experiments. When an experiment is done using 3 independent variables, it will be referred as a *three factor ANOVA*.

To display the results, we used the following standard notation: F is the statistical value and p the significance level of the results. F is computed as

$$F(x, y) = \frac{\text{Effect} + \text{Error}}{\text{Error}}$$

with x , the degrees of freedom (number of scores in the sample that are independent and free to vary) between treatments and y the degrees of freedom of the error. Hence, if $F = 1$, there is no effect detected. The higher the value of F , the more important the effect.

In addition, for the results to be reliable, it is necessary to compute how much of the value of F is due to an actual effect compared to the experimental error. This is obtained by computing p , the probability that F is observed due to experimental error. For the results to be significant, p needs to be lower than 0.05 which means that the results have 95% reliability. The lower the p , the more reliable the results.

For more information, the reader may consult [Gra08a].

A generic topological framework for physical simulation

Elsa Fléchon^a, Florence Zara^a, Guillaume Damiand^a, Fabrice Jaillet^{a,b}

^aUniversité de Lyon, CNRS, Université Lyon 1, LIRIS, UMR5205, F-69100, Villeurbanne, France

^bUniversité de Lyon, IUT Lyon 1, Computer Science Department, F-01000, Bourg-en-Bresse, France
firstname.name@liris.cnrs.fr

ABSTRACT

This paper presents the use of a topological model to simulate a soft body deformation based on a Mass-Spring System. We provide a generic framework which can integrate any kind of geometrical meshes (hexahedral or tetrahedral elements), using several numerical integration schemes (Euler semi-implicit or implicit). This framework naturally allows topological changes in the simulated object during the animation. Our model is based on the 3D Linear Cell Complex topological model (itself based on a 3D combinatorial map), adding the extra information required for simulation purposes. Moreover, we present some adaptations performed on this data structure to fit our simulation requirements, and to allow efficient cutting or piercing in a 3D object.

Keywords

Physically-based simulation; Mass-Spring System; Topological model; Linear Cell Complex; Hexahedral and tetrahedral elements mesh; Deformation; Topological changes; Cutting; Piercing.

1 INTRODUCTION

Following the increasing demand of realism in computer graphics, physically-based simulation has become a very active research field over the last decade. This is particularly apparent in medical simulation, interactive entertainment, and more generally in all virtual reality applications where animation, interaction or alteration of deformable objects is required in interactive time. Two perennial challenges in this domain are real-time simulation of object undergoing user's interaction like cutting, tearing or fracture, and the adaptive mesh coarsening and refinement, to better handle interaction within a simulation scene (for example in collision and contact zones).

The use of a topological model naturally provides a framework to describe objects subdivided into cells (vertices, edges, faces, volumes) and to modify their topology by performing the appropriate operations. In that case, the information required by the simulation has to be associated with cells. Moreover, an efficient implementation of this kind of data structure should minimize the impact on computation time and should still enable real-time user interactions during the simulation.

The aim of this paper is to put forward the benefits of the use of a topological model for a physical animation based on a Mass-Spring System (denoted MSS). In this sense, we use the 3D Linear Cell Complex (denoted LCC) as topological model (itself based on a 3D combinatorial map). We add on the LCC all information necessary for simulation purposes. Furthermore, we present the adaptations operated on this data structure to facilitate topology changes during the animation, and specifically to allow cutting or piercing of the simulated 3D object. This process is illustrated in Fig. 1.

The use of a topological model for a physical animation presents two main interests. (1) It describes all the cells and all the adjacent and incident relationships between these cells. This is particularly important to associate information to some cells, and to efficiently update this information during the operations. (2) It proposes rigorous operations allowing topological changes while guaranteeing the validity of the mesh. These two features make the topological model unavoidable for a problematic of adaptive mesh refinement.

In this paper, our main contributions are:

- a fully generic framework for topology-based modeling, not only to describe the relationships between geometrical elements, but specifically fitted to the context of physically-based simulation;
- an embedded structure dedicated to storing the mechanical properties, leading to facilitating all the topological changes during animation while preserving the mechanical behavior of the altered object;
- a stable and robust implementation, as the optimized structure will minimize the memory usage and at the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

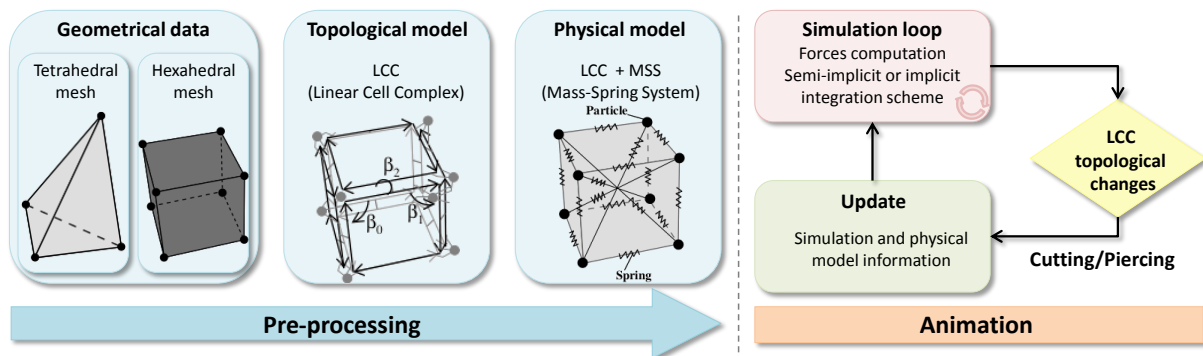


Figure 1: Overview of our method.

same time will permit a simplification of the algorithmic aspects of manipulating elements, *i.e.* low cost browsing through all elements, or accessing to neighbors;

- the detailed formulation of the force differentials (required for the Euler implicit integration scheme) in the context of soft body simulation with MSSs.

2 RELATED WORK

Topological model. Several topological models have been featured in recent years, but only a few of the proposed solutions are fully generic. The combinatorial map is such a solution [1]. It consists of a combinatorial data structure allowing a description of nD objects subdivided into cells (vertices, edges, faces, volumes) using only one basic element, called *dart*, and a set of pointers between these darts. Thanks to these pointers, all the incidence and adjacency relationships between the cells of the subdivision may be easily retrieved.

The main interest of combinatorial maps is: (1) to be generic *i.e.* defined in any dimension; (2) to fully describe the incidence and adjacency relationships between cells, which information is useful for algorithmic aspects; (3) to be possibly customized by adding any type of information to cells. For all these reasons, these models and their variants have already proven to be useful and efficient data structures for image representation and processing [2], or for geometrical modeling [3].

Physical simulation and cutting. In Computer Graphics, significant efforts have been put in proposing efficient methods to model deformable objects, as stated in the following state of the art [4]. Among them, the Finite Element Method (denoted FEM) is the most common. However, it generally requires expensive pre-computation to allow interactive topological changes. In [5], a cutting method based on a co-rotational implicit FEM [6] is presented to overcome this limitation, by successively removing, subdividing and adding elements. This avoids the reconstruction of the global matrix, but may generate ill-conditioned elements that are prone to produce numerical instabilities.

The MSS is an interesting alternative for physical modeling. Indeed, as for the tensor-mass model, it supports modification of its geometry in a more natural way, that can be managed locally. Moreover, the principle of splitting faces, instead of removing elements, will result in more plausible animations. The difficulty is then to redistribute the mass and physical parameters over the elements, but none of the past attempts was able to preserve exactly the same behavior as the original MSS. Concerning cutting, some process have been proposed to follow a cutting path either by refining the element or by moving artfully the vertices [7, 8]. This generally implies complex topological modifications and an exhaustive knowledge on the incident and adjacency relationships. For example, an algorithm is proposed in [9] to guarantee the manifold property of the object after topological changes to avoid ill-structured elements. However, this solution requires to handle substantial number of cases. Hence, in our paper, we show a cutting method along edges, particularly tailored for simulation of deformable objects.

Hybrid model. The use of a topological model to dynamically handle changes of object during a physical simulation (generally for cutting purposes) is rare enough to be worthy of note. In 2010, Meseure [10, 11] have presented in this sense a physical simulation based on a MSS using generalized maps, a variant of combinatorial maps. In 2011, Darles [12] has also used the generalized map topological model, this time for simulation based on a model of mass-interaction.

In the first two papers [10, 11], the mechanical information of the objects discretized into tetrahedra is embedded in the topological model by attaching it to the darts. Then, a semi-implicit integration scheme is used for the simulation. But some drawbacks remain in this proposition. Springs are only associated with edges of the topological model. Thus, this cannot be generalized to others geometries, as it will be impossible for example, to add inner diagonal springs in a hexahedron. There is no direct access to particles and springs. Consequently, the topological data structure is computationally expen-

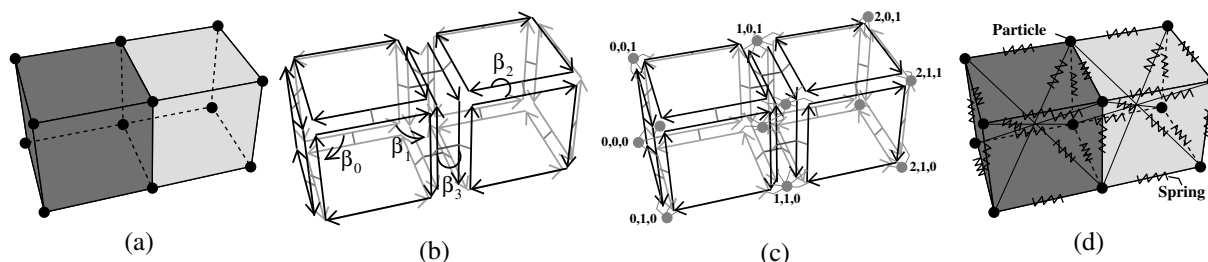


Figure 2: (a) A 3D object composed of two hexahedra (mesh). (b) The corresponding 3D combinatorial map. (c) We add 3D points linked with 0-cell to obtain a 3D LCC. (d) We associate the structures of `Particle` and `Spring` to obtain our LCC+MSS, a 3D LCC allowing a physical simulation based on a MSS.

sive just to retrieve these elements. Thus, they propose to set an additional array containing all the elements plus a pointer through the corresponding dart. This solution improves the speed up of the method, but it leads to a more complex structure and burdens significantly updating operations, as these arrays must be recomputed after each topological modification made on the generalized maps.

The solution introduced in our paper is based on similar ideas as in [10, 11], overcoming the above-mentioned limitations thanks to a more generic model (not only tetrahedra but any type of cells may be taken into account) allowing fully integrated physical simulation (currently based on a MSS) with incremental topological changes. Lastly, high level C++ mechanisms present in CGAL LCC are used. For example, functors are automatically called when a particle is split, allowing us to simplify the updating of mechanical information associated with the topological model.

3 3D LCC FOR MSS

MSSs have largely been used in animation. It consists of discretizing the object in a set of particles (also called masses) connected together by springs. The data structure of our MSS simulation is based on the 3D LCC [13] from the CGAL Open Source geometric algorithms library [14]. This structure allows a representation of an orientable 3D object subdivided into cells with linear geometry.

Fig. 2 illustrates the main steps to construct a 3D LCC for a MSS (denoted LCC+MSS): given the geometrical input data of the 3D object - Fig. 2(a), we first describe its topology with a 3D combinatorial map - Fig. 2(b) which describes cells as well as their incidence and adjacency relationships. Then, we add coordinates of the points to obtain a 3D LCC - Fig. 2(c) which describes the geometry of the objects. Finally, we associate the structure `Particle` with vertices and `Spring` with edges to fit the simulation requirements for a MSS - Fig. 2(d).

3D combinatorial map. We give here an intuitive presentation of combinatorial maps. The interested reader may find all the mathematical background in [1, 15].

In practice, a 3D combinatorial map is an edge-centered data structure composed of a set of basic elements called *darts*, four *pointers* between these darts noted $\beta_0, \beta_1, \beta_2$ and β_3 , and some constraints defined on these pointers to guarantee the topological validity of the described objects.

Fig. 2(a) presents a 3D object composed of 2 3-cells (volumes), 11 2-cells (faces), 20 1-cells (edges) and 12 points associated with the 12 0-cells (vertices). Fig. 2(b) shows the 3D combinatorial map describing this object. Each cell is described by a set of darts. Indeed, each dart (drawn by oriented segments) of the 3D combinatorial maps belongs to a vertex, an edge, a face and a volume. Each cube is described by 24 darts. Given a dart d , $\beta_1(d)$ gives the next dart belonging to the same face and volume and $\beta_0(d)$ gives the previous one; $\beta_2(d)$ gives the other dart belonging to the same edge and volume but not to the same face and $\beta_3(d)$ gives the other dart belonging to the same face and edge but not to the same volume.

Thanks to these rules, starting from a dart, the different pointers can be used to retrieve all the darts that describe the same i -cell, $\forall i \in \{0, 1, 2, 3\}$. Moreover, the adjacency and incidence relationships between the cells are entirely given by these pointers.

This basic description (darts, pointers and constraints) defines a 3D combinatorial map carrying no additional knowledge. But, some information is generally required to describe the geometry of the objects, their colors, the area of their faces, etc. Consequently, in 3D combinatorial maps, any type of information may be associated with any cell. This is done through an association between all the darts that belong to a same cell and an *attribute* containing the information. This allows a direct access to every attribute of a given dart.

3D LCC. The 3D LCC uses the mechanism of attributes to associate a 3D point with each vertex of the combinatorial map (represented by grey dots in Fig. 2(c)). Thus, the geometry of each edge of a 3D LCC is a segment whose endpoints are associated with the two vertices of the edge; the geometry of each face is obtained from all the segments associated with the edges describing the boundary of the facet; and so on.

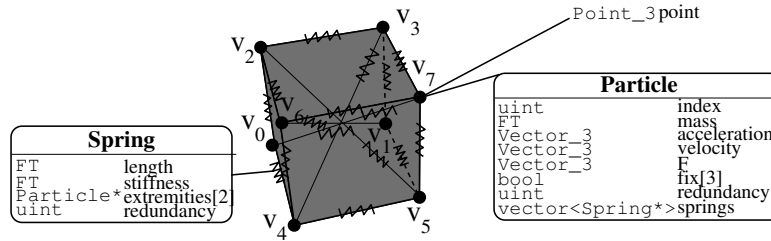


Figure 3: A 3D LCC with additional information of a physical model based on a MSS.

One strength of the CGAL implementation of the LCC is to store every attributes in a compact container which is a set of arrays. Thus, we can directly iterate through all the attributes associated with a given dimension without iterating through darts. Moreover, each array can be processed in parallel, as they are independent. Lastly, the ability to automatically call a function (defined by the user), when two attributes are merged or when an attribute is split in two during an operation, is a powerful mechanism that we use to define our cutting operation (explained in the following).

Another particularly interesting feature is that different geometrical kernels are provided within CGAL, allowing for example the use of exact or inexact arithmetic, exact or inexact geometric predicates. Each kernel defines basic types and geometric primitives. `FT` is the field number type used as basic type to define each coordinate of points and related measures. Depending on the kernel, `FT` could be `double` or an arbitrary precision number type based on the GMP library [16]. Using `FT`, `Point_3` describes a 3D point, and `Vector_3` a 3D vector. As LCC is templated by a kernel, all our code is generic and any type of kernel may be chosen without having to modify anything other than the template argument. This will allow us to easily perform tests of robustness thanks to exact arithmetic to detect possible errors coming from roundness problems.

3D LCC+MSS. To understand our topological model for a MSS, we first briefly recall here the dynamics of a MSS. Thus, considering a MSS composed of n 3D particles, we note \mathbf{M} the diagonal mass matrix (of size $3n \times 3n$), \mathbf{F} , \mathbf{V} , \mathbf{P} respectively the force, velocity and position vectors of particles (of size $3n$). For each particle i , we note m_i its mass, $\mathbf{P}_i(t)$, $\mathbf{V}_i(t)$ its position and velocity at time t , and $\mathbf{F}_i(t)$ the sum of forces it undergoes (spring forces and external forces like gravity or interaction). For each spring connecting particles i and j , we note k_{ij} its stiffness constant and l_{ij} its initial length.

Then, the dynamics of the model are governed by Newton's law with the following relation at time t for each particle i :

$$m_i \frac{d^2}{dt^2} \mathbf{P}_i(t) = \mathbf{F}_i(t). \quad (1)$$

From this equation, the acceleration of the particles may be deduced according to applied forces. A numerical integration scheme is then used to obtain velocity (according to acceleration) and position (according to velocity) of the particles.

In the current version of our framework, 3D objects are discretized either into hexahedra or tetrahedra. For a tetrahedral discretization, a particle is associated with each vertex, and respectively a spring with each edge of the object. In a same way, for a hexahedral discretization (see example in Fig. 2(d)), a particle is associated with each vertex, a spring with each edge, but in addition there are 4 internal diagonal springs for each hexahedron (with no corresponding edges, 1-cell).

Fig. 3 illustrates with more details the use of a 3D LCC to describe a MSS containing one hexahedron. It contains 8 particles (numbered from v_0 to v_7), 16 springs corresponding to the 12 edges and the 4 internal diagonals of the hexahedron.

For each 0-cell, we store the structure `Particle` corresponding to the information related to the MSS: its index (used for the Euler implicit integration scheme explained later), its mass, its acceleration, its velocity, the sum of the forces applied on this particle, and its direction constraints (to fix the position of a particle along the different axis). We also store the redundancy of the considered particle (the number of volumes incident to the vertex), and the diagonal springs linked to the considered particle, as an array of pointers to `Spring` (only for hexahedral element). Note that the position of the particle is given by the `Point_3` associated to the corresponding vertex in the 3D LCC.

For each 1-cell, we store the associated `Spring` of the MSS: its initial length, its stiffness and an array of pointers to its two extremities (*i.e.* pointers on the two `Particle` connected by the considered spring). We also store information concerning its redundancy (the number of volumes incident to the edge).

Lastly, we store the global stiffness matrix (used for the Euler implicit scheme) into the 3D LCC+MSS object as this matrix is shared by all the volumes of the same 3D LCC.

4 TOPOLOGY-BASED SIMULATION

Fig. 1 presents an overview of our method which proposes a topology-based model embedding geometrical and physical information for an efficient response to complex simulation cases. Our method is divided in two parts. (1) A pre-processing part with the construction and the initialization of our model based on the 3D LCC integrating the input parameters of the object (geometrical data, mechanical parameters). (2) An animation part with the simulation loop including the computation of the forces applied on the particles and the numerical integration scheme used to update the velocity and position of the particles; and the possibility of dynamically cutting or piercing our object during the animation. Moreover, we use either the semi-implicit or the implicit version of Euler's integration scheme, leading to fast and stable simulation.

Initialization. The first step of the topology-based simulation consists of the computation of the mechanical information, and in the initialization of the attributes associated with the 3D LCC.

We start by reading an input file describing the geometrical information of the hexahedral or the tetrahedral mesh, and we create the corresponding elementary volumes in the 3D LCC. Then, we identify all the faces that share the same geometry to obtain a connected object.

Next, we iterate through all the particles (0-cells) to initialize the mass, redundancy and index. (1) The index is initialized at the creation of a new particle. (2) The redundancy is the number of volumes incident to the vertex, which can be directly computed using the incidence relations. (3) For the computation of the mass of each particle, the global mass of the object is properly distributed over its n particles. Consequently, considering a 3D homogeneous object discretized into hexahedra with a mass density ρ , the mass m_i of each particle i of the MSS is defined by:

$$m_i = \sum_{j|i \in E_j} \frac{\rho V_{E_j}}{8}$$

with E_j the set of hexahedra of volume V_{E_j} containing the particle i . Identically for a tetrahedral discretization, we get:

$$m_i = \sum_{j|i \in E_j} \frac{\rho V_{E_j}}{4}$$

with E_j the set of tetrahedra of volume V_{E_j} containing the particle i . Note that it is also possible to attribute a specific mass density to each element, in case of heterogeneous material.

Then, we iterate through all the edges (1-cells) of the LCC to create and initialize the corresponding springs. (1) The stiffness constant is calculated and attributed to each spring of the MSS accordingly to the desired

Young modulus E and the Poisson ratio ν (in our examples, we set $E = 100$ MPa and $\nu = 0.3$). Our calculations are based on the formulations given in [17, 18]. (2) The redundancy is the number of volumes incident to the edge, which again can be directly computed. (3) We initialize also the two extremity pointers by using the incidence relationships given by the LCC.

Lastly, we iterate through all the volumes (3-cells) to create diagonal springs when necessary. For each hexahedron, we create the four inner diagonal springs, initialize their information, and insert them into the corresponding particles.

Forces computation. The first step of the simulation's loop is to compute all the forces applied on the particles, due to springs or external interactions. The force involved at time t by a spring connecting particles i and j is defined by:

$$\mathbf{F}_{ij}(t) = \mathbf{F}_{ij}^e(t) + \mathbf{F}_{ij}^v(t) \quad (2)$$

- $\mathbf{F}_{ij}^e(t)$ is the elasticity force of this spring defined by:

$$\begin{cases} \mathbf{F}_{ij}^e(t) &= k_{ij} (d_{ij} - l_{ij}) \mathbf{U}_{ij}(t) \\ \mathbf{F}_{ji}^e(t) &= -\mathbf{F}_{ij}^e(t) \end{cases}$$

with $d_{ij} = \|\mathbf{P}_j(t) - \mathbf{P}_i(t)\|$ and $\mathbf{U}_{ij}(t)$ the normalized direction vector defined as:

$$\mathbf{U}_{ij}(t) = \frac{\mathbf{P}_j(t) - \mathbf{P}_i(t)}{\|\mathbf{P}_j(t) - \mathbf{P}_i(t)\|}$$

- $\mathbf{F}_{ij}^v(t)$ is the viscosity force of this spring (used to simulate dissipated energy due to frictions) defined by:

$$\mathbf{F}_{ij}^v(t) = \gamma_{ij} [(\mathbf{V}_j(t) - \mathbf{V}_i(t)) \cdot \mathbf{U}_{ij}(t)] \mathbf{U}_{ij}(t)$$

with the spring's viscosity coefficient defined by [19]:

$$\gamma_{ij} = 2\sqrt{\frac{m_i + m_j}{2}} k_{ij}$$

To enable this computation with our 3D LCC+MSS model, we first iterate through all the 0-cell attributes to reset the particles force to a null vector, and we add the external forces. In this work we only consider the gravity by adding $m_i \mathbf{g}$ to \mathbf{F}_i (with $\mathbf{g} = 9.8$ m/s²). Then, we iterate through each spring (both non-diagonal and diagonal for hexahedral elements). We compute the force of the considering spring by using equation (2) and we accumulate the calculated force on the two particles linked to it. All the required information is stored in the topological model and may be directly accessed through the attributes (the position, velocity and force of the particles; the extremities, initial length, redundancy, stiffness of the springs).

Numerical integration schemes. The second step of the simulation's loop is to compute the velocity and position of all the particles by using a numerical integration scheme. This enables the update of the geometrical coordinates of all the 0-cells of the LCC.

The *Euler semi-implicit* integration method has been implemented first. After the computation of the acceleration of all the particles (using equation (1)), we get for a time step h :

$$\begin{cases} \frac{d}{dt}\mathbf{P}_i(t+h) &= \frac{d}{dt}\mathbf{P}_i(t) + h\frac{d^2}{dt^2}\mathbf{P}_i(t) \\ \mathbf{P}_i(t+h) &= \mathbf{P}_i(t) + h\frac{d}{dt}\mathbf{P}_i(t+h) \end{cases}$$

But, to obtain a stable simulation, h has to be reduced, especially when stiffness increases. So, to enable larger time steps, the *Euler implicit scheme* has been implemented:

$$\begin{cases} \frac{d}{dt}\mathbf{P}_i(t+h) &= \frac{d}{dt}\mathbf{P}_i(t) + h\frac{d^2}{dt^2}\mathbf{P}_i(t+h) \\ \mathbf{P}_i(t+h) &= \mathbf{P}_i(t) + h\frac{d}{dt}\mathbf{P}_i(t+h) \end{cases}$$

This scheme may be reformulated as follows [20]:

$$\left(\mathbf{M} - h\frac{\partial\mathbf{F}(t)}{\partial\mathbf{V}(t)} - h^2\frac{\partial\mathbf{F}(t)}{\partial\mathbf{P}(t)}\right)\Delta\mathbf{V} = h\mathbf{F}(t) + h^2\frac{\partial\mathbf{F}(t)}{\partial\mathbf{P}(t)}\mathbf{V}(t)$$

with

$$\Delta\mathbf{V} = \frac{d}{dt}\mathbf{P}(t+h) - \frac{d}{dt}\mathbf{P}(t)$$

and $\partial\mathbf{F}/\partial\mathbf{P}$, $\partial\mathbf{F}/\partial\mathbf{V}$, the Jacobian matrices (of size $3n \times 3n$, for n particles) encoding the variation of forces resulting from position and velocity change at time t . After computing these matrices, this linear system is solved using the Conjugate Gradient method to obtain $\Delta\mathbf{V}$. Then, the velocity is updated with:

$$\frac{d}{dt}\mathbf{P}(t+h) = \frac{d}{dt}\mathbf{P}(t) + \Delta\mathbf{V}$$

and the position with:

$$\mathbf{P}(t+h) = \mathbf{P}(t) + h\frac{d}{dt}\mathbf{P}(t+h)$$

Naturally, the damping coming from the environment is taken into account when updating the velocity of the particles. Moreover, we can note that the acceleration is never really computed according to equation (1) within the Euler implicit scheme.

Computation of the global stiffness matrix $\partial\mathbf{F}/\partial\mathbf{P}$. To fill, at time t , the nonzero entries of matrix $\partial\mathbf{F}/\partial\mathbf{P}$, we compute $\partial\mathbf{F}_{ij}/\partial\mathbf{P}_i$ (matrix of size 3×3) only if particles i and j are connected (with $i \neq j$), with:

$$\frac{\partial\mathbf{F}_{ij}}{\partial\mathbf{P}_i} = \frac{\partial\mathbf{F}_{ij}^e}{\partial\mathbf{P}_i} + \frac{\partial\mathbf{F}_{ij}^v}{\partial\mathbf{P}_i}$$

with

$$\begin{cases} \frac{\partial\mathbf{F}_{ij}^e}{\partial\mathbf{P}_i} = k_{ij} \left[\frac{l_{ij}}{d_{ij}} (I - \mathbf{U}_{ij} \mathbf{U}_{ij}^T) - I \right] \\ \frac{\partial\mathbf{F}_{ij}^v}{\partial\mathbf{P}_i} = \frac{\gamma_{ij}}{d_{ij}} [[\mathbf{U}_{ij} \mathbf{U}_{ij}^T - I] \omega + [W \mathbf{U}_{ij} \mathbf{U}_{ij}^T]^T] \end{cases}$$

where I is the identity matrix of size 3×3 and W a diagonal matrix defined by:

$$W = -\frac{\mathbf{V}_{ij}}{\mathbf{U}_{ij}} + \omega \times (1, 1, 1)^T \text{ with } \omega = \mathbf{V}_{ij} \cdot \mathbf{U}_{ij}$$

Fig. 4 illustrates the matrix $\partial\mathbf{F}/\partial\mathbf{P}$ for a 2D MSS with 6 particles. This matrix is composed of:

- non-diagonal elements $\partial\mathbf{F}_{ji}/\partial\mathbf{P}_i$ and $\partial\mathbf{F}_{ij}/\partial\mathbf{P}_j$ deduced from $\partial\mathbf{F}_{ij}/\partial\mathbf{P}_i$ with:

$$\frac{\partial\mathbf{F}_{ij}^{e/v}}{\partial\mathbf{P}_i} = -\frac{\partial\mathbf{F}_{ji}^{e/v}}{\partial\mathbf{P}_i}, \quad \frac{\partial\mathbf{F}_{ji}^{e/v}}{\partial\mathbf{P}_i} = \frac{\partial\mathbf{F}_{ij}^{e/v}}{\partial\mathbf{P}_j}$$

- diagonal elements $[\partial\mathbf{F}/\partial\mathbf{P}]_{i,i}$ with:

$$\left[\frac{\partial\mathbf{F}}{\partial\mathbf{P}}\right]_{i,i} = \sum_{k \in S} \frac{\partial\mathbf{F}_{ik}}{\partial\mathbf{P}_i}$$

with S the set of particles connected to i .

Note that this global stiffness matrix is filled using the particle index.

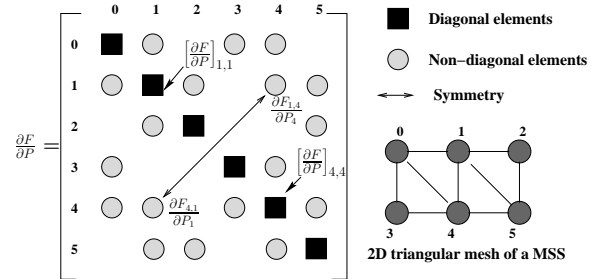


Figure 4: Illustration of the Jacobian matrix $\partial\mathbf{F}/\partial\mathbf{P}$ for a 2D MSS composed of 6 particles.

Computation of the damping matrix $\partial\mathbf{F}/\partial\mathbf{V}$. The matrix $\partial\mathbf{F}/\partial\mathbf{V}$ corresponds to the Rayleigh Damping defined by $\mu M + \lambda \partial\mathbf{F}/\partial\mathbf{P}$, with μ and λ the mass and stiffness proportional Rayleigh damping coefficients.

5 TOPOLOGICAL CUTTING

Amongst all the changes that a 3D object may undergo during an animation, cutting is one of the most challenging, as it implies deep topological changes, *i.e.* element removing or splitting. Without an efficient topological model, it may cause difficulties, as generating

ill-structured and non-manifold mesh when refining elements or moving points to follow the cutting path. In this section, we demonstrate through this particular operation that the proposed model is robust and well adapted, with limited additional computational cost.

Unsewing in 3D LCC. In a 3D LCC, the i -unsew operation, $i \in \{1, 2, 3\}$, unglues two i -cells which are glued along one of their $(i-1)$ -cells. For that, we unlink β_i pointers for all the darts belonging to the shared $(i-1)$ -cell. After this operation, the initial shared $(i-1)$ -cell is split in two $(i-1)$ -cells, and respectively this initial $(i-1)$ -cell will no more be shared by the two i -cells.

If the unsew operation splits a j -cell c , $\forall j \in \{0, 1, 2, 3\}$, in two j -cells $c1$ and $c2$, and if c is associated with a j -attribute `attr1`, then this attribute is duplicated into `attr2`, and all the darts belonging to $c2$ are associated with this new attribute. Next, a functor is called on the two attributes allowing the user to update its specific information.

In Fig. 5, an example of 3-unsew operation is presented. We start from the LCC given in Fig. 5(a) made up of 4 hexahedra. A 3-unsew operation is processed to unglue the dark grey and the white hexahedra. The face separating these two hexahedra (named (v_1, v_2, v_3, v_4) in the initial configuration) is split in two by the 3-unsew operation. We can see in Fig. 5(b) that this split involves the duplication of the two vertices v_1 and v_2 , and the duplication of the three edges (v_4, v_1) , (v_1, v_2) and (v_2, v_3) . The user defined functor is called on each pair of duplicated cells, for example on (v_1, v'_1) for vertices, and $((v_4, v_1), (v_4, v'_1))$ for edges. In this example, note that vertices v_3 and v_4 are not duplicated during the unsew operation as they are still connected by the two grey hexahedra right below, and consequently the edge (v_3, v_4) is not duplicated either.

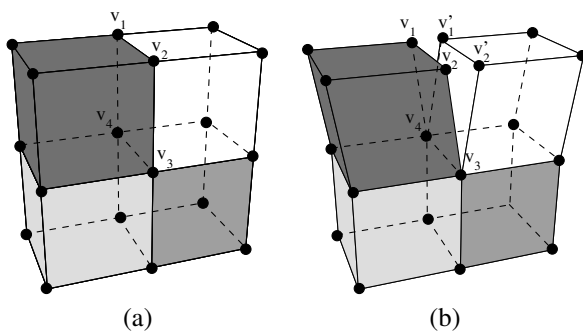


Figure 5: Example of 3-unsew in a 3D LCC.

Object cutting during simulation. During the physical simulation, we use the 3-unsew operation to cut an object by separating two adjacent volumes. As explained above, this operation duplicates the attributes where necessary. In our case, particles and springs are

possibly duplicated. In both cases, the physical information stored in the attributes has to be updated.

For new particle. If the cutting involves the creation of a new particle (for example particle v'_1 in Fig. 5), all the information is first duplicated from the initial particle (v_1). Then we initialize the index of v'_1 to a new index (used for the Euler implicit integration) and we update the mass of v_1 and v'_1 by decrementing their redundancy (*i.e.* the number of volumes incident to the particles). Lastly, we update the list of diagonal springs connected to v_1 and v'_1 . Indeed, springs associated with the second volume are still attached to the initial particle v_1 , that is incorrect (see Fig. 6(a)). Thus, we iterate through all the springs associated with v_1 and for each one whose other extremity belongs to the second volume, we detach it from v_1 and attach it to v'_1 (see Fig. 6(b)).

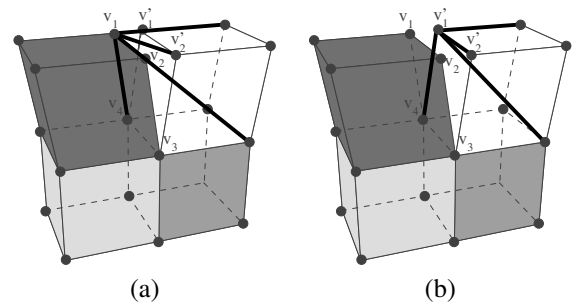


Figure 6: Modification of springs in the 3D LCC+MSS after 3-unsewing. (a) The bold segments represent the springs wrongly attached to particle v_1 before updating. (b) Same springs after updating.

For new spring. If the cutting involves the creation of a new spring (for example edge (v'_1, v'_2) in Fig. 5), all the information of the spring associated with the initial edge (here, the edge (v_1, v_2)) is duplicated on the new spring created and associated with the new edge. Then, we update the redundancy of the two springs by counting the number of volumes incident to each corresponding edge, and we update the two extremities of the new spring to link the two particles incident to the new edge.

As a consequence of the creation of new particles, the cutting of the object involves the re-sizing of the data structures that store the mechanical information: force, acceleration (for Euler's semi-implicit scheme), velocity, position vectors, and the global stiffness matrix (for Euler's implicit scheme). Then, the computation of the forces applied on each particle is performed as usual.

6 RESULTS

In this section, we present some results to validate the behavior of our animation based on a topological model. We show how our model can simulate deformation of a soft body and how it can be cut or pierced

during the animation. All our experiments were carried out on an Intel Core i7 processor (2.40 GHz with 4 multi-threaded cores). We set the material properties to $E = 100$ MPa, $\nu = 0.3$ and $\rho = 1,000$ kg/m³ to simulate behaviors similar to soft tissues. All the given times correspond to one step of the simulation process, in milliseconds. For all the presented results, only the gravity force is applied. Moreover, the red spheres in figures are particles constrained in all directions and the 3-unsewed faces are drawn in red.

Semi-implicit vs. implicit integration scheme. We simulate a cube modeled by 1 hexahedron with 16 springs and 8 particles, and the same cube modeled with 5 tetrahedra, 18 springs and 8 particles. Table 1 compares the computation times resulting from the use of the Euler semi-implicit and implicit integration schemes (with $h = 1$ ms). As expected, the implicit one is slower, but this will be advantageously counter-balanced by more stability, allowing larger time steps.

	Semi-implicit	Implicit
Hexahedral mesh	0.009	0.42
Tetrahedral mesh	0.012	0.42

Table 1: Time (in ms) per simulation step.

Scale up. Fig. 7 presents the scale up property of our simulation by considering a beam with an increasing number of hexahedral elements: 1 cube of 10 cm; $2 \times 2 \times 2$ cubes of 5 cm; $4 \times 4 \times 4$ cubes of 2.5 cm; $8 \times 8 \times 8$ cubes of 1.25 cm; $16 \times 16 \times 16$ cubes of 0.625 cm (simulations made with $h = 0.1$ ms) and $32 \times 32 \times 32$ cubes of 0.3125 cm (simulation made with $h = 0.01$ ms). Results show that the complexity of our method is linear according to the degree of freedom (DOF) within our system.

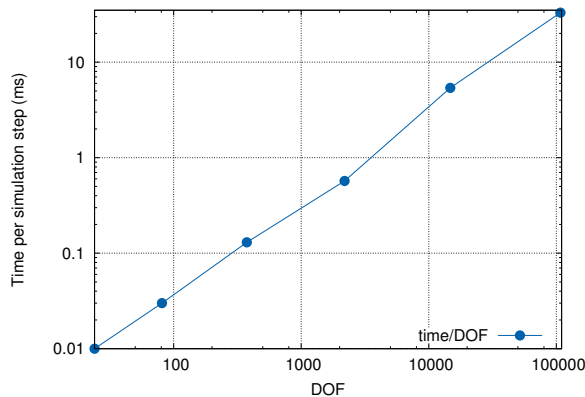


Figure 7: Time (in ms) per simulation step according to the size of a hexahedral beam (with log-log scale).

Comparison performance with SOFA. We now consider a mesh with 640 hexahedra, 4,954 springs and 891 particles. We compare our simulation times with the ones obtained using our same MSS implemented in the Open Source Framework SOFA [21, 22]. With our topological framework, we obtain an average of

0.55 ms by simulation step with the Euler semi-implicit integration scheme, while we get 0.77 ms by using the Euler explicit integration scheme in SOFA (with $h = 1$ ms). We note that the additional cost due to the topological structure remains limited, using a similar integration scheme. This first comparison is very encouraging as it shows that our method is competitive even in its preliminary form (not fully optimized).

Comparison with another topological model. In [11] a similar solution has been proposed. With a mesh composed of 1,856 tetrahedra, 2,850 springs and 564 particles, they stated that the simulation step takes 8 ms on a dual core 2.8GHz processor, using an Euler semi-implicit integration (4th order Runge-Kutta integration scheme).

For comparison purposes, we built a similar beam composed of 1,890 tetrahedra, 2,655 springs and 480 particles. We obtain an average of 0.5 ms by simulation step when using the Euler semi-implicit integration (with $h = 1$ ms). Even if these results do not involve the same CPU, nor exactly the same mesh, this preliminary comparison is very satisfactory and demonstrates that the proposed structure is well adapted for simulation applications, leading to minimizing the additional memory and operative cost induced by the topological model.

Cutting. As presented above, cutting objects during the animation is easily supported by the proposed structure. In the following, the Euler implicit integration scheme is used for its stability when high deformation is undergone by the 3D object.

Fig. 8, 10 and 11 show examples of interactive deformation and cutting of a beam. The user is provided with tools permitting him to select the particles belonging to the face to 3-unsew. If necessary, a zone can be selected by the user, where all the faces including the selected particles are 3-unsew.

In Fig. 9, we present the cutting performed on two bigger meshes representing a frog. This illustrates that our method can be used for detailed objects.

Piercing. Thanks to our cutting method, we can pierce an object by 3-unsewing all the faces around the volume to be removed. In Fig. 12, we pierce a beam composed of $5 \times 3 \times 3$ hexahedral elements by 3-unsewing 3 cubes in the middle of the beam.

7 CONCLUSION AND PERSPECTIVE

In this paper, we have presented the use of the 3D LCC topological model for a physical simulation based on a MSS. The mechanical information of the object is added to the data structure as attributes associated with i -cells, avoiding the duplication of data and the management of different data structures. Our first experiments illustrate that our method is competitive, even without any specific optimization.

In the future, we first plan to improve our simulation time. All the operations used for the simulation can be easily implemented in parallel, and a version of the simulation on the GPU is under investigation. Moreover, we work to insert other physical models like tensor-mass model, finite element method or even a more evolved mass-spring system. Finally, we want to integrate in our framework the automatic refining and coarsening of cells during the simulation. Indeed, the 3D LCC provides all the basic topological operations required for that kind of operation. Consequently, the next stage will be: how to update the physical information and to define the criteria to decide where these operations must be applied? This improvement will allow us to overcome the current limitation of cutting only along the faces of the mesh. Indeed thanks to the subdivision features, we will be able to follow a cutting path through elements.

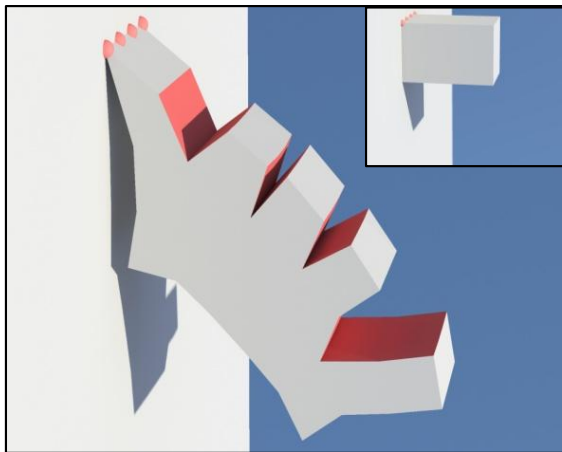


Figure 8: 10 faces are 3-unsewed of a beam composed of $5 \times 3 \times 3$ elements (initial state in top right).

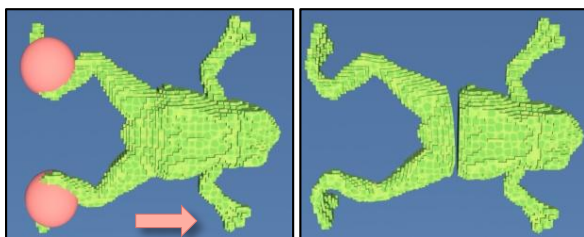


Figure 9: Cutting one hexahedral mesh representing a frog with 26,125 hexahedra and 32,934 particles.

ACKNOWLEDGEMENT

The authors would like to thank Eric Galin for his helpful comments. This work was performed within the framework of the LabEx PRIMES (ANR-11-LABX-0063) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR); and partially funded by the ANR-MN project SAGA (ANR-12-MONU-0006).

8 REFERENCES

- [1] P. Lienhardt. Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Comput. Aided Des.*, 23(1):59–82, 1991.
- [2] G. Damiand. Topological model for 3D image representation: Definition and incremental extraction algorithm. *Computer Vision and Image Understanding*, 109(3):260–289, March 2008.
- [3] D. Fradin, D. Meneveaux, and P. Lienhardt. A hierarchical topology-based model for handling complex indoor scenes. *Computer Graphics Forum*, 25(2):149–162, June 2006.
- [4] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.
- [5] H. Courtecuisse, H. Jung, J. Allard, C. Duriez, D. Y. Lee, and S. Cotin. GPU-based real-time soft tissue deformation with cutting and haptic feedback. *Progress in Biophysics and Molecular Biology*, 103(2-3):159–168, 2010. Special Issue on Soft Tissue Modelling.
- [6] M. Nesme, Y. Payan, and F. Faure. Efficient, Physically Plausible Finite Elements. In *Eurographics'05 (short papers)*, Dublin (IRL), 2005.
- [7] B. Lee, D. C. Popescu, and S. Ourselin. Topology modification for surgical simulation using precomputed finite element models based on linear elasticity. *Progress in Biophysics and Molecular Biology*, 103(2-3):236–251, 2010. Special Issue on Biomechanical Modelling of Soft Tissue Motion.
- [8] C. Dick, J. Georgii, and R. Westermann. A hexahedral multigrid approach for simulating cuts in deformable objects. *IEEE TVCG*, 17(11):1663–1675, 2011.
- [9] C. Forest, H. Delingette, and N. Ayache. Removing Tetrahedra from manifold tetrahedralisation : application to real-time surgical simulation. *Medical Image Analysis*, 9(2):113–122, 2005.
- [10] P. Meseure, E. Darles, and X. Skapin. A Topology-Based Mass/Spring System. In *Proc. of CASA'2010 (short papers)*, St Malo (F), June 2010.
- [11] P. Meseure, E. Darles, and X. Skapin. Topology-based Physical Simulation. In *Proc. of VRIPHYS'10*, pages 1–10, Copenhagen (DK), November 2010.
- [12] E. Darles, S. Kalantari, X. Skapin, B. Crespin, and A. Luciani. Hybrid physical-topological modeling of physical shapes transformations. In *Proc. of DMDCM'11*, pages 154–157, Washington, DC (USA), 2011.
- [13] G. Damiand. Linear Cell Complex. In *CGAL User and Reference Manual*. CGAL Editorial Board. <http://www.cgal.org/Pkg/LinearCellComplex>.
- [14] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.2 edition. <http://www.cgal.org/>.
- [15] G. Damiand. Combinatorial maps. In *CGAL User and Reference Manual*. CGAL Editorial Board. <http://www.cgal.org/Pkg/CombinatorialMaps>.

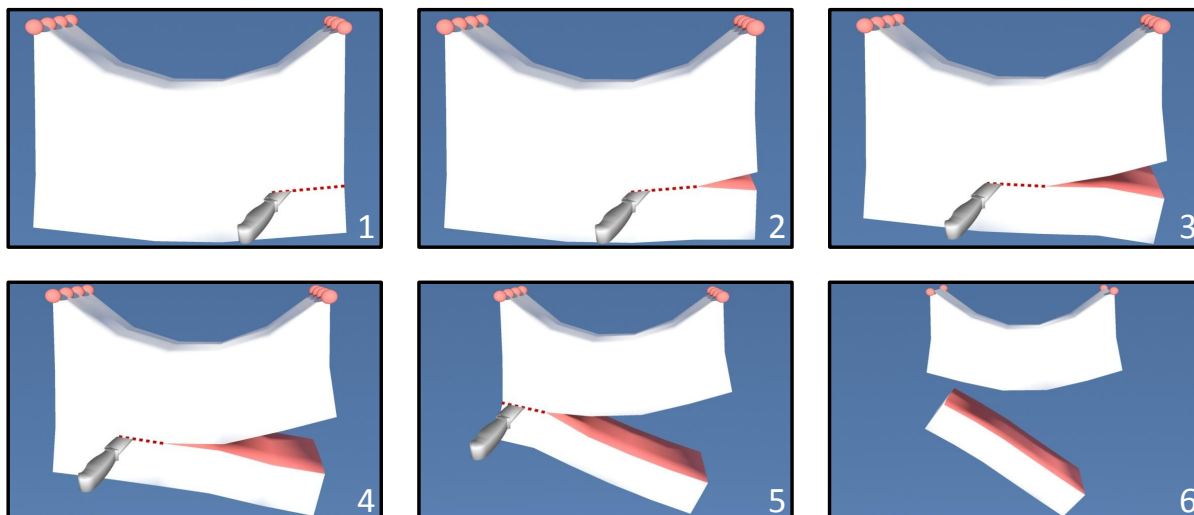


Figure 10: Beam composed of $5 \times 3 \times 3$ elements, progressively cut by a knife (from the Avalon 3D archive).

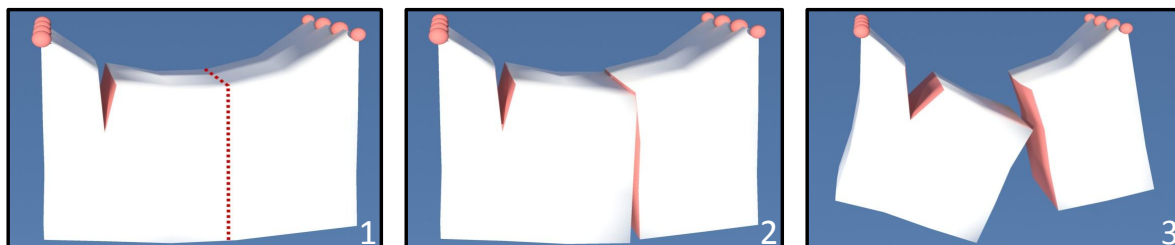


Figure 11: Beam composed of $5 \times 3 \times 3$ elements. 1) 3 faces are 3-unsewed. 2) and 3) Two states after the 3-unsewing of 9 faces.

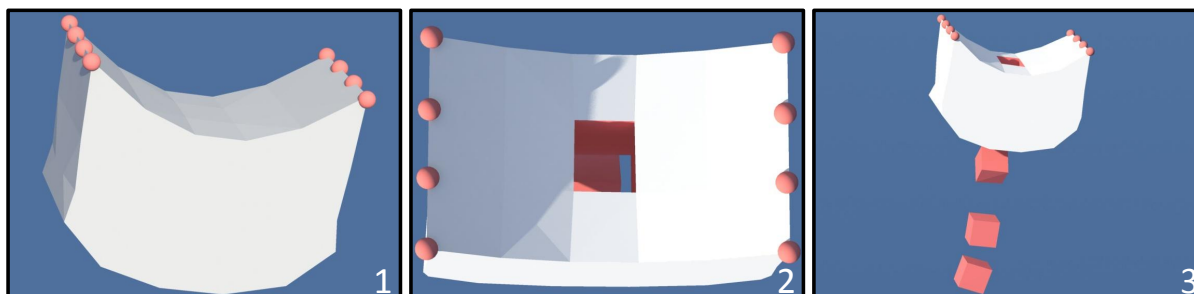


Figure 12: Beam composed of $5 \times 3 \times 3$ hexahedral elements. 1) Initial deformed state. 2) and 3) Two views of the same object after piercing by 3-unsewing 14 faces.

- [16] Gmp library : <http://gmplib.org>.
- [17] V. Baudet, M. Beuve, F. Jaillet, B. Shariat, and F. Zara. Integrating Tensile Parameters in Hexahedral Mass-Spring System for Simulation. In *Proc. of WSCG'09*, pages 145–152, February 2009.
- [18] V. Baudet, M. Beuve, F. Jaillet, B. Shariat, and F. Zara. Integrating Tensile Parameters in Mass-Spring System for Deformable Object Simulation. Technical report, LIRIS UMR CNRS 5205, September 2009.
- [19] Y. Bhasin and A. Liu. Bounds for damping that guarantee stability in mass-spring systems. *Studies in health technology and informatics*, 119:55–60, 2005.
- [20] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proc. of the 25th annual conference on Computer Graphics and Interactive Techniques*, pages 43–54. ACM, 1998.
- [21] J. Allard, S. Cotin, F. Faure, P.-J. Bensoussan, F. Poyer, C. Duriez, H. Delingette, and L. Grisoni. SOFA - an Open Source Framework for Medical Simulation. In *MMVR 15*, pages 13–18, Palm Beach, USA, 2007.
- [22] Sofa : <http://www.sofa-framework.org>.

Interactive Grass Rendering Using Real-Time Tessellation

Klemens Jahrmann

Vienna University of
Technology

klemens.jahrmann@net1220.at

Michael Wimmer

Vienna University of
Technology

wimmer@cg.tuwien.ac.at

ABSTRACT

Grass rendering is needed for many outdoor scenes, but for real-time applications, rendering each blade of grass as geometry has been too expensive so far. This is why grass is most often drawn as a texture mapped onto the ground or grass patches rendered as transparent billboard quads. Recent approaches use geometry for blades that are near the camera and flat geometry for rendering further away. In this paper, we present a technique which is capable of rendering whole grass fields in real time as geometry by exploiting the capabilities of the tessellation shader. Each single blade of grass is rendered as a two-dimensional tessellated quad facing its own random direction. This enables each blade of grass to be influenced by wind and to interact with its environment. In order to adapt the grass field to the current scene, special textures are developed which encode on the one hand the density and height of the grass and on the other hand its look and composition.

Keywords

Rendering, Grass, Blades, Tessellation, Geometry, Wind, Real-time, LoD

1 INTRODUCTION

Grass rendering plays an important role for rendering outdoor scenes. Especially in virtual reality and computer games, natural scenes should include properly rendered vegetation. For trees and plants, many solutions already exist, but rendering grass has been a problem for a long time due to the high amount of needed geometry. This is why most often grass is rendered as a texture mapped to the ground or as transparent billboard quads. Both approaches have different unwanted artifacts: grass as a texture does not look good if it is shown from a small angle since there are no displacements of the ground, while billboards have problems when they are viewed from above because then they look very flat.

In this paper, we present a technique which is capable of rendering whole grass fields in real time completely as geometry. This can be achieved by using the tessellation shader as a fast geometry enhancement tool and some level-of-detail approaches, see Figure 1. The technique also uses special textures like density and terrain maps to pass important information to the tessellation stage.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.



Figure 1: Screenshot from our system.

2 STATE OF THE ART

Early grass rendering methods use only flat textures mapped onto the ground. A more sophisticated texture-based method was proposed by Shah et al. [6]. It uses a bidirectional texture function to animate the grass texture and displacement mapping for silhouettes, which produces a better look than a simple texture, but needs more time to render. Until today, almost every method still uses a texture-based technique for rendering grass that is far away from the camera. For grass that is closer to the viewer, different approaches have been developed: Orthmann et al. [5] present a grass-rendering technique which uses two billboards to represent a bunch of grass and which can interact with its environment. Rendering billboards can be done very efficiently on the graphics card but looks flat when viewed from higher angles. Also, billboards lack visual depth. To get deeper looking grass, Habel et al. [3] use half-transparent texture slices, which are placed

on a regular grid, but the visual artifacts for higher angles still exist. In order to solve this problem, Neyret [4] refers to existing fur rendering techniques using 3D textures and extends them to render high-detail geometry for large scenes like grass. This method allows him to render grass with good quality from different angles but still looks artificial. Zhao et al. [8] use the geometry shader to draw bunches of grass as geometry, but due to performance reasons, the generated grass field is very sparse.

Like in this paper, Wang et al. [7] render a whole grass field using geometry by drawing a single blade of grass multiple times. The blade is modeled with several vertices and a skeleton for the animation, but complex level-of-detail and culling approaches have to be implemented to achieve reasonable framerates for rather sparse grass fields. Boulanger et al. [1] present a hybrid technique which uses the techniques mentioned before as different levels of detail. As geometric representation, they use a single grass patch consisting of blades of grass which are formed by the trajectories of particles during a preprocessing step. To get a large grass field, instanced rendering is used with a random orientation of the patch for each instance to reduce the grid artifacts which come with instancing. Grass that is further away is rendered using a 3D texture-based technique to draw a set of vertical and horizontal texture slices. For rendering high distances, only one horizontal slice is rendered, which leads to a texture mapped onto the ground.

3 OVERVIEW

In contrast to the methods mentioned before, the proposed technique renders *all* blades of a dense grass field completely as smoothly shaped geometry using hardware tessellation. In addition, each blade of grass can be individually deformed by forces like wind or objects.

First, we explain a basic technique suitable for smaller grass fields. The particular point of this technique is that it stores the geometric data of each blade of grass individually. This leads to a perfectly random and uniform grass field and has good performance, because many level-of-detail approaches can be applied to it. However, since each blade is stored by the application it does not scale very good for large scenes.

We then extend the basic technique using instancing: only a single grass patch is stored in memory and drawn multiple times, so that animated grass scenes of practically arbitrary sizes can be rendered in real time. A common handicap of instancing are the repeating patterns that normally appear, but since each blade is generated inside the shader, its look can be easily influenced by the world-space position, which conceals the transitions between the patches. In addition, by using

instancing, we can simply add billboard flowers to the scene for visual enhancement.

Both methods require an initialization step to set up the textures (Section 4). We discuss the basic rendering technique in Section 5, and extensions to handle larger scenes, including instancing, levels of detail and antialiasing in Section 6.

4 INITIALIZATION

Our technique is based on a grass field which forms the boundaries of the space where grass is rendered. The field itself acts as a container for user-specified data like textures and parameters and is represented as a regular grid, where each grid cell contains a grass patch. The size of each grid cell has a significant impact on the performance, but the optimal size cannot be predefined in general, since it depends heavily on the overall scene and application. Each grass patch saves a vertex array object, which contains all necessary data to draw the blades of grass that grow on it. In addition, a bounding box is assigned to the patch to perform view-frustum culling.

The visual representation of the rendered grass field can be adjusted to match a high variety of possible scenes. It is determined by various textures, explained in Section 4.1, and by several parameters, which are listed below. During the initialization step, both the textures and parameters are evaluated to generate the input for the rendering process.

- *Density*: indicates how many blades of grass are initialized on a 1x1 unit-sized square.
- *Width*: the minimum and maximum width of a blade.
- *Height*: the minimum and maximum height of a blade.
- *Bending factor*: determines the maximum bending offset.
- *Smoothness factor*: specifies the maximum tessellation of a blade.

4.1 Special Textures

4.1.1 Density Map

The density map defines the density and the height over a grass field. Its red channel encodes the density value, which is directly used by the application to determine how many blades are generated. The green channel gives a subsampled version of the grass, which is later used when a less detailed model is needed. For example, when rendering water reflections, only the geometry near the coast lines need to be rendered. The blue

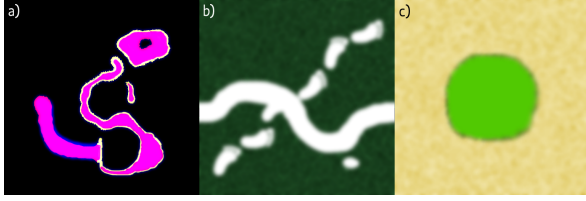


Figure 2: a) Shows an example of a density map, b) illustrates how a terrain map can be used (the white pixels are transparent) and c) shows a vegetation map of an oasis scene.

channel encodes the blades of grass' height at each position. If no special height differences are desired, the blue component can be created by blurring the red channel, which leads to a smoother transition between the regions where grass is rendered and where it is not. An example of a density map can be seen in Figure 2a.

4.1.2 Terrain Map

The terrain map itself does not change the appearance of a grass blade, but it determines the texture underneath the grass, which is also very important, since grass is naturally not perfectly dense and it is seen especially at sparse spots or tracks. Basically, the ground beneath the grass is organized in tiles just like the grass. Each tile has its own seamless diffuse texture assigned and the terrain map works as diffuse texture for the whole field. How a ground fragment is shaded is then determined by the alpha value of the terrain map. This can be seen in Equation 1, where c_r is the result color, c_{tt} the sampled tile texture, c_{tm} the sampled terrain map and a its alpha value. An example of a terrain map can be seen in Figure 2b.

$$c_r = c_{tt} \cdot (1 - a) + c_{tm} \cdot a \quad (1)$$

4.1.3 Vegetation Map

A vegetation map is used if the grass color depends on its position inside a scene. If it is enabled, the color of the grass is determined by the color sampled from the vegetation map rather than from its diffuse texture. This can be used to simulate a desert scene with an oasis, where lush and healthy grass is placed near the water and dry grass further away. An example of a vegetation map can be seen in Figure 2c.

4.2 Creating Geometry

At the end of the initialization step, when all user-specified data has been read, the basic grass geometry that will serve as input to the rendering pipeline is generated for the high- and the low-detailed version of the grass field. For this, the density map is sampled and for each pixel, the world-space area which it refers to is calculated. The area is then multiplied with the red sample

and the density value to determine how many blades have to be generated at this specific area. Then for each blade a random position is calculated and added to the high-detail list. If the density map's green sample value is also greater than zero, it is added to the low-detail list too. For each blade, an outline quad is generated with the following vertices:

$$\begin{aligned} w &= w_{\min} + R \cdot (w_{\max} - w_{\min}) \\ h &= (h_{\min} + R \cdot (h_{\max} - h_{\min})) \cdot \text{density}_b \\ \mathbf{p}_1 &= \mathbf{p}_c - [0.5 \cdot w, 0.0, 0.0] \\ \mathbf{p}_2 &= \mathbf{p}_c + [0.5 \cdot w, 0.0, 0.0] \\ \mathbf{p}_3 &= \mathbf{p}_c + [0.5 \cdot w, h, 0.0] \\ \mathbf{p}_4 &= \mathbf{p}_c - [0.5 \cdot w, h, 0.0] \end{aligned} \quad (2)$$

where w is the width and h the height of the blade with their maxima and minima w_{\max} , w_{\min} , h_{\max} and h_{\min} , R is a randomly generated number between zero and one, density_b is the blue component sampled from the density map, \mathbf{p}_c is the blade's center point and \mathbf{p}_i is the i^{th} point of the quad. For higher performance, the height map can also be baked into the position, to reduce the texture lookups during the rendering process.

Apart from the vertex position, each vertex sends additional information to the graphics card: two-dimensional texture coordinates and the blade's center position, with a y-value of zero if it is a lower vertex or one if it is an upper vertex. Up to now, each blade of grass differs only in its height and its position. To make each blade unique, random values have to be generated and added to the graphics pipeline. For this, each of the transferred vectors is filled with random values until they are all four-dimensional, and an additional vector containing only random values is passed too. In total, there is one random value for the position and center position vector, two values for the texture coordinates and four additional ones, resulting in eight random values, which will all be needed in the rendering process.

5 REAL-TIME GENERATION

The rendering of the grass is done completely by the graphics card, which has to be able to execute tessellation shaders. If a grass patch is visible inside the view frustum, all included blades of the desired detail are drawn. Since each blade of grass is just a two-dimensional quad, backface-culling has to be disabled for full visual appearance.

5.1 Grass-Blade Generation

At first, each blade of grass is represented just by a random sized quad aligned on the x - y -plane, which is then tessellated into subquads using the tessellation

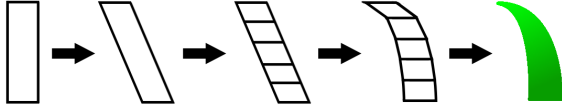


Figure 3: Illustration of the creation of a single blade of grass. The first step shows the input quad, the following steps show the results after the vertex shader, the tessellation control shader, the tessellation evaluation shader and finally the shaded blade of grass after the fragment shader.

shader. The resulting vertices are aligned on two parallel splines, which are formed by the upper and lower vertices of the input quad and two additional control points that determine the curvature of a blade. The final shape of the blade is determined by an alpha texture. The following will describe the path of the quad through each shader stage to become a nicely formed and shaded blade of grass. The procedure is also illustrated in Figure 3.

During the vertex shader stage, the blade's orientation is specified by applying a rotation around the blade's center position with one of the random values as angle. For the upper vertices, which are identified by their center position's y -value, an offset in x - z -direction is applied by using two other random numbers R_1, R_2 multiplied by the maximal bending factor b specified for the current grass patch. The calculation of the offset can be seen in Equation 3.

$$\text{offset} = \begin{bmatrix} b \cdot (2R_1 - 1) \\ 0 \\ b \cdot (2R_2 - 1) \end{bmatrix} \quad (3)$$

In the tessellation control shader, the distance between the blade and the camera position is computed. Using this distance together with a specified maximum distance, the tessellation factor is calculated by linearly interpolating the integer values between 0 and the given smoothness parameter s , which indicates the maximum tessellation of the blade. The interpolation can be seen in Equation 4, where d describes the distance to the camera and d_{\max} the maximum distance.

$$\text{level} = \left\lceil s \cdot \left(1 - \frac{d}{d_{\max}} \right) \right\rceil \quad (4)$$

This value is applied as inner tessellation level and also as outer tessellation level for the right and left side of the quad. At and above the maximum distance, the tessellation level of zero culls a blade completely, and near the camera, the tessellation level of s subdivides the quad to s quads along the y -axis. An alternative approach for determining the tessellation level is to calculate the screen-space size of a blade and setting the

level so that every subquad occupies the same amount of pixels. This is shown in Equation 5, where \mathbf{u}_y and \mathbf{l}_y indicate the y -coordinate of the upper and lower vertices of the quad, h_s is the vertical screen resolution and s is the smoothness parameter.

$$\text{level} = \min \left(\frac{((\mathbf{u}_y \cdot 0.5 + 0.5) - (\mathbf{l}_y \cdot 0.5 + 0.5)) \cdot h_s}{\# \text{ pixels per quad}}, s \right) \quad (5)$$

Two additional control points are generated in the tessellation control shader to determine the shape of the blade in the next step. For this, the lower points' x - and z -coordinate together with the upper y -coordinate can simply be taken, but in order to generate slightly differently shaped blades of grass, the coordinates are also varied by two random numbers. The variation can be seen in Equation 6, where $\mathbf{l} = (\mathbf{l}_x, \mathbf{l}_y, \mathbf{l}_z)$ and $\mathbf{u} = (\mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z)$ describe the lower and upper vertex and R_i are random values. In our application, R_1 lies between $-\frac{1}{4}$ and $\frac{1}{4}$ and R_2 between $\frac{3}{4}$ and $\frac{5}{4}$.

$$\mathbf{h} = \begin{bmatrix} \mathbf{l}_x \cdot R_1 + \mathbf{u}_x \cdot (1 - R_1) \\ \mathbf{l}_y \cdot R_2 + \mathbf{u}_y \cdot (1 - R_2) \\ \mathbf{l}_z \cdot R_1 + \mathbf{u}_z \cdot (1 - R_1) \end{bmatrix} \quad (6)$$

In the tessellation evaluation shader, the tessellated quad is shaped by aligning the vertices along two parallel quadratic splines. Each spline is defined by three control points. We use De Casteljau's algorithm [2] to compute each desired curve points $c(v)$, with v being the domain coordinate (equals i/level for the i^{th} subquad) as parameter. The evaluation of one vertex on one spline can be seen in Equation 7 and is also illustrated in Figure 4a, where \mathbf{p}_b and \mathbf{p}_t indicate the bottom and top vertices and \mathbf{h} is the additional control point. By using this recursive approach, the tangent \vec{t} of a spline can also be found easily.

$$\begin{aligned} \mathbf{a} &= \mathbf{p}_b + v \cdot (\mathbf{h} - \mathbf{p}_b) \\ \mathbf{b} &= \mathbf{h} + v \cdot (\mathbf{p}_t - \mathbf{h}) \\ \mathbf{c}(v) &= \mathbf{a} + v \cdot (\mathbf{b} - \mathbf{a}) \\ \vec{t} &= \frac{\mathbf{b} - \mathbf{a}}{\|\mathbf{b} - \mathbf{a}\|} \end{aligned} \quad (7)$$

When both splines are computed, the final position and tangent can be interpolated using the domain coordinate v as parameter. The bitangent results directly from the two spline points. After tangent and bitangent are calculated, the normal can be computed by the cross product. This can be seen in Equation 8 and is also illustrated in Figure 4b.

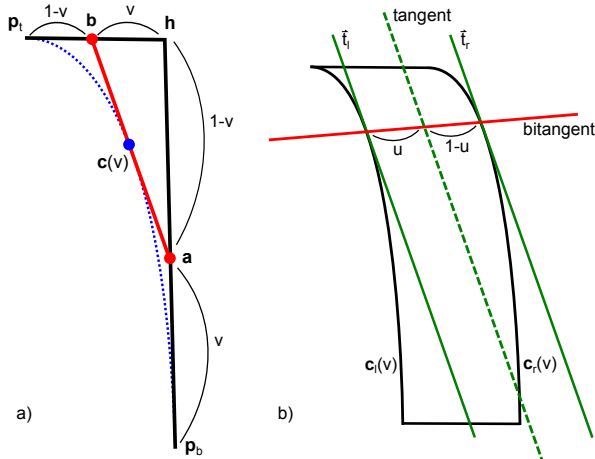


Figure 4: a) Illustration of De Casteljau's algorithm for calculating a point on a curve with the parameter v . b) shows how the tangent and bitangent of the blade of grass can be calculated.



Figure 5: Shows an example of an alpha texture (top) and a diffuse texture (bottom).

$$\begin{aligned}
 \text{position} &= \mathbf{c}_l(v) \cdot (1 - u) + \mathbf{c}_r(v) \cdot u \\
 \text{bitangent} &= \frac{\mathbf{c}_r(v) - \mathbf{c}_l(v)}{\|\mathbf{c}_r(v) - \mathbf{c}_l(v)\mathbf{a}\|} \\
 \text{tangent} &= \frac{\vec{t}_l \cdot (1 - u) + \vec{t}_r \cdot u}{\|\vec{t}_l \cdot (1 - u) + \vec{t}_r \cdot u\|} \\
 \text{normal} &= \frac{\text{tangent} \times \text{bitangent}}{\|\text{tangent} \times \text{bitangent}\|}
 \end{aligned} \quad (8)$$

In the fragment shader, the final shape of the blade is formed by the masking using the alpha texture. The diffuse color is then sampled either by the blade's diffuse texture or by the vegetation texture as has been explained in Section 4.1.3. In order to achieve a better variance, the components of the sampled color are modified slightly by three random values. Due to shadow effects, a blade of grass is normally darker near the ground and lighter at its tip, which can be simulated by multiplying the color with the vertical texture coordinate, leading to a better looking parallax effect. An example of an alpha and diffuse texture is shown in Figure 5.

5.2 Applying Forces

Until now, the grass is rendered nicely, but it is not animated and does not interact with its environment. In nature, grass is always in motion, either influenced by wind or by objects which also leave tracks. Wind can

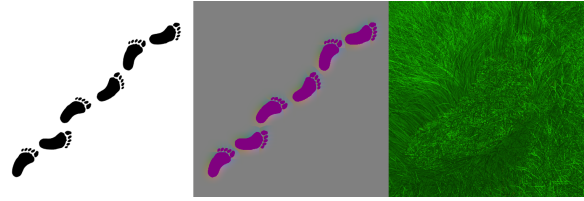


Figure 6: The left image shows an image of foot prints, the middle illustrates its corresponding force map and the right shows the rendered grass scene with the force map applied.

either be applied through a texture which is computed in some application-specific way, e.g. by a physical simulation, or it can be calculated using a wind function inside the shader. We implemented a wind function $w(\mathbf{p})$ that takes the world space position $\mathbf{p} = (\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z)$ and the current time t as parameter and calculates wind as two overlapping sine and cosine waves along the x -axis and a cosine wave along the z -axis. The function can be seen in Equation 9, where the constants c_i determine the shape of the wind waves together with a small number ε to avoid a division by zero. The result of the wind function can then be applied directly to the offset of the quad's upper vertices in the vertex shader.

$$\begin{aligned}
 w(\mathbf{p}) &= \sin(c_1 \cdot a(\mathbf{p})) \cdot \cos(c_3 \cdot a(\mathbf{p})) \\
 a(\mathbf{p}) &= \pi \cdot \mathbf{p}_x + t + \frac{\pi}{4} \frac{1}{|\cos(c_2 \cdot \pi \cdot \mathbf{p}_z)| + \varepsilon}
 \end{aligned} \quad (9)$$

In addition to wind, each blade of grass can be influenced individually by external forces. For this, we use a force map which indicates the direction in which a blade is pushed at a certain position. In case of simulating tracks, the vectors of areas which have been fully under pressure by an object point towards the negative y -direction. At the boundary of an object, the force vectors point away from the object's center point, which can be approximated by the normal vector at the boundary. The vectors of the regions around an object also point away from its center, but with decreasing length. The size of the surrounding region depends on the impact of the applied force. An example can be seen in Figure 6. The sampled force-map vector can then be added to the offset of the quad's upper vertices.

6 HANDLING LARGE SCENES

6.1 Visibility

When dealing with large scenes, not all grass patches are visible in each frame. So the application has to determine which patches can be seen by a camera frustum, so that only visible blades of grass are transmitted to the graphics card. For this we implemented a simple hierarchical rasterization approach, where every two patches

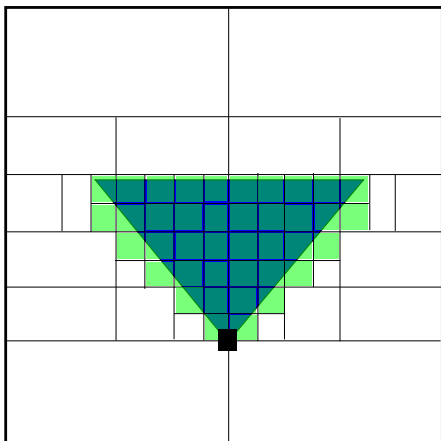


Figure 7: Illustration of the visibility test for the hierarchical rasterization approach.

are combined to one bigger patch until the grass field is represented by only one macro-patch. Then each frame the macro-patch is tested for visibility with the view frustum and if it is visible, its subpatches are tested iteratively. Figure 7 shows the visibility test for the hierarchical rasterization approach. The boxes representing the big patches are tested against the blue frustum and the green quads indicate the visible grass patches.

6.2 Instancing

As mentioned in Section 3, storing each single blade of grass in memory has hard limits regarding the maximum scale of a scene. Therefore, we implemented an instanced rendering technique, which only needs to store a single patch of grass as geometry data, in order to minimize the memory overhead of the scale of a scene.

When drawing grass patches by instancing, the basic grass-rendering method stays the same, but some adaptations have to be done. The grass field still needs to have a maximum size in order to generate lookup coordinates for the various global textures, but a tiling factor can be applied so that the textures do not need to scale with the scene if they are seamless. Instead of a list of patches, the grass field has to create just a single patch. The appropriate transformation matrices for the visible patch positions are calculated during the rasterization of visible patches. Then before each draw call, the transformation matrices have to be transmitted to the graphics card.

In the rendering process, only the vertex shader stage has to be changed. The vertex position has to be transformed by the given transformation matrix, and the density value has to be tested if the blade is visible. In addition, the height map has to be sampled, since it cannot be baked into the position anymore. These changes lead to an additional transformation and two additional

texture lookups for each vertex of each blade, which is a certain overhead over the basic technique for smaller scenes.

Normally, when using instancing on patches, a repeating grid pattern appears over the scene, which is very disturbing. This effect can be partly hidden by rotating or mirroring the instances, like it is done by Boulanger et al. [1]. Since in our method, the blades are generated inside the shaders, the world-space position can be taken directly as variation factor of each random value R so that no grid pattern is visible without any rotation or mirroring. The influence of the world-space position on a random value can be described by one of the following Equations 10-12, where 11 as well as 12 can also be calculated using the z -coordinate or in 12, the cosine function can be used too. In these equations, $\mathbf{p} = (\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z)$ is the world-space position, $\text{dimension} = (\text{dimension}_x, \text{dimension}_y)$ refers to the size of the grass field, c stands for any constant value and the function $\text{fract}()$ gives the fractional part of a number.

$$R_{\text{new}} = \text{fract}\left(\frac{R \cdot \mathbf{p}_x}{\mathbf{p}_z}\right) \quad (10)$$

$$R_{\text{new}} = R \cdot \frac{\mathbf{p}_x}{\text{dimension}_x} \quad (11)$$

$$R_{\text{new}} = R \cdot \sin(c \cdot \mathbf{p}_x) \quad (12)$$

6.3 Level-of-Detail

Drawing each single blade of grass each frame has some disadvantages: First of all, it leads to a low framerate, and secondly, aliasing artifacts become visible the more blades of grass are drawn onto the same pixel. So several level-of-detail approaches have to be implemented to overcome these problems. As mentioned in Section 4, a highly downsampled version of a grass patch is generated for additional effects like reflections. Following this approach, more versions can be calculated at startup, each with fewer blades of grass. Then, for rendering, the distance to the grass patch determines the detail which is drawn.

This will then lead to step artifacts when the distance falls beyond a detail threshold. To achieve smooth transitions, more random blades of grass have to be culled the higher the distance gets. This takes place in the tessellation control shader, where the tessellation level is set to zero for some blades depending on one of its random values and the ratio between distance and maximum distance. Since the blades are normally distributed, hardly any artifacts are visible. In addition, the tessellation level of a blade of grass gets smaller the further away it is, because at far distances, there is no difference if the geometry of a blade is just a single quad.

In order to reduce the maximum amount of blades that are drawn, each blade that lies beyond a given maximal distance is culled by the tessellation control shader. This will result in a hard-edged circle of grass around the camera position. A smoother transition can be achieved by lowering the height of a blade of grass the further it is away. For these distance-driven approaches different distance functions were tested during the research process, and surprisingly a linear function yields the best results regarding performance and appearance.

6.4 Antialiasing

Although fewer blades of grass are drawn at higher distances, there are still aliasing and z-fighting artifacts visible. The reason is that it is impossible to render the perfect amount of blades for all distances and viewing directions. Nevertheless, the artifacts can be hidden by blurring the result image in consideration of the pixel's depth value. To maintain real-time performance, we use a separable 7×7 blur kernel with half-pixel sampling for a screen resolution of 1200×800 . Since OpenGL's depth values go from one at the near clipping plane to zero at the far clipping plane, the depth value can be taken directly as center weight for the kernel function. The other kernel weights can either be the same small value, which leads to a simple average filter for higher distances, or small gradient values, leading to a Gaussian-like filter. The blur will then be processed in two separate passes, the first in horizontal and the second in vertical direction. For each pass, five samples are taken, the first at the pixel's center and the other at the border between the neighboring pixel and its neighbor, so that for a 7×7 kernel, only ten texture lookups are needed.

Blurring the image can reduce the aliasing artifacts only to a certain amount, but at farther distances, high-frequency noise can still be seen. So we developed another approach, which applies a penalty to the material of a blade of grass depending on its distance to the camera. This leads to a smooth darkening towards the horizon, which not only reduces the aliasing a lot, but also delivers a good sense of depth of a scene. Together with the distance blur, the artifacts can be reduced to a pleasing amount. To illustrate the impact of these effects, Figure 8 shows different result images.

7 RESULTS

For the performance evaluation, three scenes have been tested: First, a small oasis scene with reflections and very dense grass around the lake as well as dry and sparse grass further away. The second scene shows a meadow with complex objects and with a force map of huge foot steps applied. The last tested scene shows a large undulating grass field with flowers and the force map of the second scene using instanced rendering.

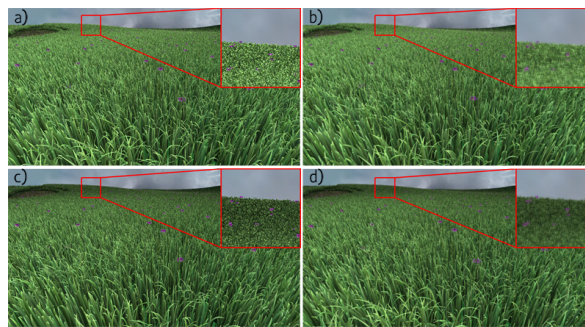


Figure 8: These result images show the effects of the distance blur and the depth darkening. Image a) is rendered with both effects disabled, b) is blurred without depth darkening, c) has depth darkening enabled but is not blurred and d) is rendered with both effects applied.

The application was tested on a Windows 7 64-bit system, with an 8-core Intel i7 processor at 3.8GHz with 16GB of RAM and an NVIDIA GeForce GTX 680 with 4GB of graphics memory.

In Table 1, the average frames-per-second together with the corresponding amount of blades drawn are measured for the three different scenes. It is tested with and without post-processing effects (the distance blur and the depth darkening) and with differently scaled grass fields. Some tests also contain additional geometry besides the grass field, which is stated in the column *Add. drawn*.

Scene	FPS	Vis. Blades	Scale	PP effects	Add. drawn
Oasis	75	136,650	160x160	enabled	-
Oasis	78	136,650	160x160	disabled	-
Oasis	65	330,816	320x320	enabled	-
Oasis	68	330,816	320x320	disabled	-
Meadow	84	348,370	300x300	enabled	-
Meadow	88	348,370	300x300	disabled	-
Meadow	83	348,370	300x300	enabled	Objects
Meadow	87	348,370	300x300	disabled	Objects
Meadow	81	340,716	600x600	enabled	Objects
Meadow	85	340,716	600x600	disabled	Objects
Instance	60	660,000	300x300	enabled	-
Instance	62	660,000	300x300	disabled	-
Instance	56	824,000	600x600	enabled	-
Instance	58	824,000	600x600	disabled	-
Instance	55	1,008,000	2,000x2,000	enabled	Flowers
Instance	57	1,008,000	2,000x2,000	disabled	Flowers
Instance	55	1,008,000	4,000x4,000	enabled	Flowers
Instance	57	1,008,000	4,000x4,000	enabled	Flowers
Instance	55	1,008,000	8,000x8,000	enabled	Flowers
Instance	57	1,008,000	8,000x8,000	enabled	Flowers
Instance	55	1,008,000	90,000x90,000	enabled	Flowers
Instance	57	1,008,000	90,000x90,000	enabled	Flowers

Table 1: Test results of three different scenes. Scene Oasis containing water reflections and a vegetation map and scene Meadow containing highly detailed objects are rendered both with the basic technique. Scene Instance is rendered using the instanced rendering approach together with billboard flowers.

For the results we tried to position the camera for worst-case scenarios where most blades of grass are drawn. The corresponding images can be seen in the

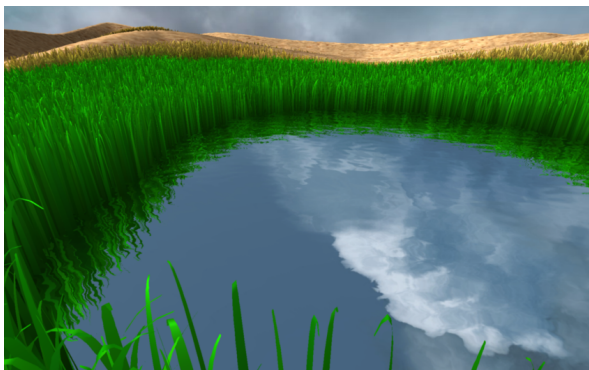


Figure 9: The rendered image of the Oasis scene.

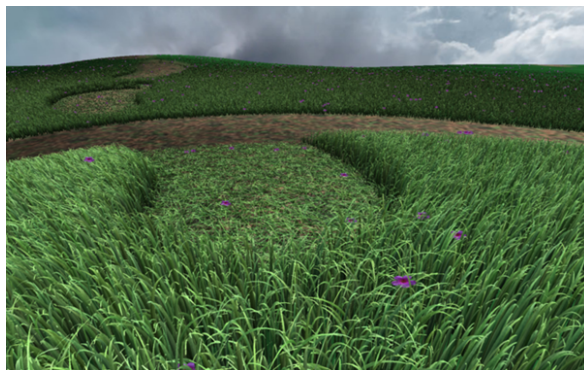


Figure 11: The rendered image of the Instance scene.



Figure 10: The rendered image of the Meadow scene.



Figure 12: The grass-rendering technique applied to a wheat field.

Figures 9, 10 and 11. As stated in earlier sections, for smaller scenes the highly optimized version is superior to the instanced version, but the big advantage of the instanced version is that its performance stays constant with the amount of blades drawn regardless of the size of the overall scene. The only limitation which the size of the scene has lies in the memory used for the grass field's hierarchical rasterization structure. The difference in the performance between the basic and the instanced method can be seen at the 300x300 and 600x600 sized instances. Even when drawing complex objects, the optimized basic version is clearly faster than the instanced version. However, the basic version is not able to render fields greater than 1000x1000, because then the application's memory is full.

A numerical comparison of the performance between the presented technique and the related papers mentioned in Chapter 2 cannot be done correctly, because most of the related techniques use static level-of-detail and image based methods, whereas our method only uses dynamic level-of-detail and geometry. The main advantage of the presented algorithm is its flexibility, so that it is extensible and can easily fit to any possible scene. This is because in each frame, the blades are procedurally generated inside the shader. With our method, also grass-like scenes like for example a wheat field can be rendered, which is shown in Figure 12. In addition,

grass rendered with the proposed technique can be used in physically correct simulations, since each blade of grass can be influenced by its environment separately.

Compared to Boulanger et al. [1], our technique enables each single blade to be animated and influenced by forces at all distances, since the grass field is completely rendered as geometry. Wang et al. [7] can also animate each single blade of grass separately, but due to the high amount of vertices which have to be stored for a scene, the grass field can only be sparse in order to achieve interactive framerates. The difference between our method and the related works can be seen in Figure 13 and Figure 14.

The mentioned scenes and effects can also be seen in motion on the accompanying video and a more detailed view of the grass and flowers can be seen in Figure 15.

8 CONCLUSION AND FUTURE WORK

Although rendering dense grass fields completely as geometry seemed to be impossible for a long time, this paper introduced a quite simple method which is able to draw large grass scenes in real time using the tessellation shaders as geometry enhancement tool. One of the main advantages is its flexibility regarding different

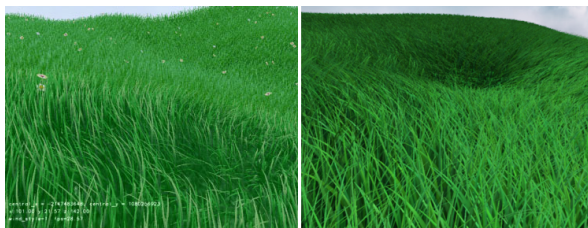


Figure 13: The left image shows a picture of a grass field with a tornado-like wind taken from [7] and the right one shows a similar scene rendered with our method, where the tornado is simulated with a forcemap.

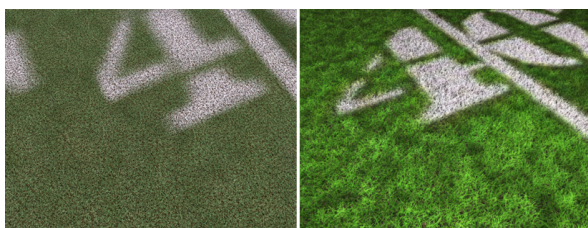


Figure 14: The left image shows a picture of a football field taken from [1] and the right one shows a similar scene rendered with our method.



Figure 15: A more detailed view of the grass and the flowers.

scenes, since the blades of grass are procedurally generated inside the shader. Following this aspect, many extensions can be made to the proposed technique. For example, a life-time model can be implemented, making grass grow over time, or the grass can adapt itself to its environment, like growing along the shape of objects. Until now the grass can only grow in y -direction, but it might be simple to give each blade of grass a direction, so that grass can grow on arbitrary locations. For example, grassy objects or roots of underground scenes can be rendered using this approach.

In the technique as presented, each blade of grass can be influenced by forces, but the animation and interaction is just a simple simulation. A physically based wind and bending model with collision detection, like in [7], has to be implemented for more realistic looking scenes.

9 REFERENCES

- [1] K. Boulanger, S.N. Pattanaik, and K. Bouatouch. Rendering Grass in Real Time with Dynamic Lighting. *Computer Graphics and Applications, IEEE*, 29(1):32–41, 2009.
- [2] Gerald E. Farin and Dianne Hansford. *The essentials of CAGD*. A K Peters, 2000.
- [3] Ralf Habel, Michael Wimmer, and Stefan Jeschke. Instant Animated Grass. *Journal of WSCG*, 15(1-3):123–128, 2007.
- [4] Fabrice Neyret. A General and Multiscale Model for Volumetric Textures. In *Graphics Interface*, pages 83–91. Canadian Human-Computer Communications Society, 1995.
- [5] J. Orthmann, C. Rezk-Salama, and A. Kolb. GPU-based Responsive Grass. *Journal of WSCG*, 17:65–72, 2009.
- [6] Musawir A. Shah, Jaakko Kontinen, and Sumanta Pattanaik. Real-time rendering of realistic-looking grass. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia, GRAPHITE '05*, pages 77–82, New York, NY, USA, 2005. ACM.
- [7] Changbo Wang, Zhangye Wang, Qi Zhou, Chengfang Song, Yu Guan, and Qunsheng Peng. Dynamic modeling and rendering of grass wagging in wind: Natural phenomena and special effects. *Comput. Animat. Virtual Worlds*, 16(3-4):377–389, July 2005.
- [8] Xiangkun Zhao, Fengxia Li, and Shouyi Zhan. Real-time animating and rendering of large scale grass scenery on gpu. In *Proceedings of the 2009 International Conference on Information Technology and Computer Science - Volume 01*, pages 601–604. IEEE Computer Society, 2009.

Analysis of Interactive Editing Operations for Out-of-Core Point-Cloud Hierarchies

Claus Scheiblauer
Vienna University of
Technology, Austria
scheiblauer@cg.tuwien.ac.at

Michael Wimmer
Vienna University of
Technology, Austria
wimmer@cg.tuwien.ac.at

ABSTRACT

In this paper we compare the time and space complexity of editing operations on two data structures which are suitable for visualizing huge point clouds. The first data structure was introduced by Scheiblauer and Wimmer [SW11] and uses only the original points from a source data set for building a level-of-detail hierarchy that can be used for rendering points clouds. The second data structure introduced by Wand et al. [WBB+07] requires additional points for the level-of-detail hierarchy and therefore needs more memory when stored on disk. Both data structures are based on an octree hierarchy and allow for deleting and inserting points. Besides analyzing and comparing these two data structures we also introduce an improvement to the points deleting algorithm for the data structure of Wand et al. [WBB+07], which thus allows for a more efficient node loading strategy during rendering.

Keywords

viewing algorithms, point clouds, complexity analysis, data structures

1 INTRODUCTION

Within the last 10 years the generation of huge point clouds with more than 10^9 points has become quite common, due to the fact that distance measurement devices like laser or triangulation scanners have increased the scanning speed and density of their sampling measurements. The latest generation of laser scanners is capable of generating point data sets consisting of 10^9 points or more with a single scan [Rie13, FAR13]. Rendering or processing point clouds of such huge sizes is still a challenge, as they usually do not fit into the main memory of normal computer workstations [WBB+07, SW11]. Even though nowadays 16GB of main memory are not uncommon, it is still not enough to handle those point clouds, as the data for the point positions alone consumes 12GB, assuming that every point position consists of three 32bit floating point numbers.

Although it is possible to split point clouds into smaller parts and work only on these, the general view for the complete data gets lost when only working with a part of the complete point cloud. Another possibility would be to downsample the original data, but then the editing operations can only be performed on a coarser resolu-

tion of the point cloud than would be available. Therefore data structures that allow to render, process, and search within huge point clouds are necessary if an overview of the whole data set is advantageous, or if tasks have to be accomplished on the original data set.

The contributions of this paper are:

- A complexity analysis and comparison of the Modifiable Nested Octree (MNO) introduced by Scheiblauer and Wimmer [SW11] and the octree-based data structure introduced by Wand et al. [WBB+07].
- An improvement to the points deleting algorithm for the data structure of Wand et al. [WBB+07], which thus allows for a more efficient node loading strategy during rendering.

We show that editing and processing operations on the points stored in these data structures are comparable in complexity to editing and processing operations on specialized data structures. These operations can be performed on out-of-core managed point clouds efficiently, if the data is managed properly with a least-recently-used (LRU) cache [Den68, Sam06].

2 PREVIOUS WORK

There exists a plethora of data structures that can be used to store and process point data sets. A structured overview of the most well known variants is given by Samet [Sam06]. Deciding on which data structure to use is dependent on the specific use case and task that

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

shall be accomplished. Each data structure is designed for a set of use cases, and one data structure is usually not the best choice for all tasks. Another distinction between the variants of the data structures are the implementation complexities of each.

A simple data structure for ordering and fast searching of points in 2D is the point quadtree. A quadtree is a hierarchical data structure where at each node the underlying space is subdivided into 4 subnodes (children) along axis-parallel lines that go through a given data point. An octree is analogous to a quadtree, but in 3D. Exactly one point is stored at each node of the hierarchy. Searching or inserting a point triggers a $O(\log(N))$ complexity search from the root node to the required node. N is the total number of points inserted. Deleting a point is quite involved, as the hierarchy has to be maintained also when points from interior nodes have been deleted, and the simplest method to do this is to rebuild the data structure from scratch.

While a point quadtree subdivides the data space of the points, other variants like the point region quadtree (PR quadtree) subdivide the space the data exists in. Nodes of the PR quadtree are subdivided at arbitrary axis-aligned lines, e.g., the ones through the center of the node, not at lines that go through given data points as in the ordinary quadtree. Another difference is that data points are only stored at leaf nodes, while interior nodes are used for searching the requested point in the proper node. Complexities for inserting and searching are dependent on the height of the PR quadtree, which in turn is dependent on the minimum distance between 2 neighboring points. To alleviate this dependency, a bucket PR quadtree can be used. In this variant of the PR quadtree up to m points are allowed in a leaf node.

If a data set is too large to fit into main memory, n -way trees can be used to access such a data set. A well known data structure for this use case is the B-tree [BM72]. Like the bucket PR quadtree it stores the data in its leaf nodes and the keys for accessing the data in its interior nodes. There are two main differences to a bucket PR tree though, first the number of children per node is not limited to 4, and second, the leaf nodes are all at the same tree level. A B-tree has only 3 to 5 levels typically [Bay08], and therefore it is a very efficient data structure for accessing points on slow secondary storage devices, because the number of accesses to the storage device is kept low. B-trees and its variants like the B+ tree or the B* tree are also used by file systems, e.g., by NTFS, the Windows file system [Mic03], to manage file accesses.

Other requirements for a data structure occur when huge point sets have to be rendered to screen. In this case the points have to be organized in a way that is easy to handle for a graphics card, but also efficient to load from disk. Ideally the points are available in

chunks that can be directly used for rendering. Another requirement is that the data structure has to support a level-of-detail (LOD) mechanism. Data structures have been proposed [GM04, PSL05] which comply to these requirements. They hold the tree access structure in memory all the time, and only load the visible points from disk, which is efficient for rendering. Editing operations on these point data structures are inefficient for different reasons. In [GM04] a uniform sampling density is assumed, and the original point set is subsampled to get the points for each hierarchy node. Adding new points, which invalidate the uniform sampling assumption, is not supported. In [PSL05] a global indexing scheme is applied to the point set which has to be updated if some points were to be removed or added.

Data structures that allow for rendering but also for efficient editing operations on out-of-core managed point clouds have been proposed [WBB+07, SW11]. We compare the editing operations of these two data structures, where Wand et al. [WBB+07] use additionally created points and Scheiblauer and Wimmer [SW11] use original points for a LOD hierarchy.

Processing data sets out-of-core can be done efficiently if the editing algorithms load chunks of data into memory before processing them, and only work on these chunks [Vit08]. Samet [Sam06] mentions that holding recently accessed chunks in a cache store could further reduce the number of disk accesses when processing data. Such cache stores are implemented in all before mentioned point rendering and editing algorithms.

3 MICHAEL WAND OCTREE RECAP

Wand et al. [WBB+07] introduced a multiresolution hierarchical data structure for rendering and manipulating huge point clouds. It is based on a region octree [Sam06], which is a data structure where the children of a node are congruent boxes regularly subdividing the space of the parent node. In the Michael Wand Octree (MWO) each interior node has inscribed a grid, and each cell of the grid can store one point. These points are used for a LOD representation of the point cloud. They are additionally created during build up of the data structure. The original points are stored in the leaf nodes. An example of a 2D MWO with 3 levels can be seen in Figure 1.

For our tests we do not use the original implementation of the point hierarchy in the XGRT system [WBB+09], but re-implemented it according to [WBB+07] in our point cloud rendering framework. Selecting points is done with a Selection Octree [SW11], which is a separate data structure describing a space inside which all points are selected.

3.1 Inserting points

Listing 1 shows how points are inserted into an MWO. Leaf nodes can only hold up to a predefined threshold

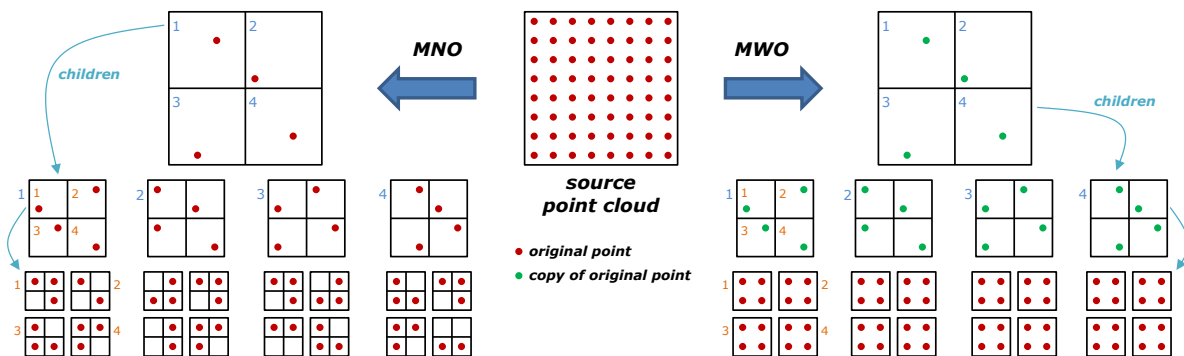


Figure 1: A source point cloud represented as MNO (left) and MWO (right). Both point hierarchies use a 2x2 grid at the interior nodes. The MNO uses the grid also at the leaf nodes, while the MWO uses an array.

m_{leaf} , e.g., 100.000 points. Each interior node holds a grid, and at each grid cell a point is stored. Furthermore a 64bit integer is maintained to count the points falling into a grid cell. For the grid we use a resolution of 2^7 cells on each of the 3 space axis, so in total a maximum of $(2^7)^3 = 128^3$ points can be stored at one interior node. Usually the number of points in an interior node is much lower for the data sets we tested.

```

foreach point P in source point data do // Main loop
  root node R->insert(P)
end

node:: insert (P): // Method
  if node is leaf
    insert P into array of points at node
    iterate ()
  else
    insert P into grid
    if grid cell GC containing P is empty
      store a copy of P in GC
      increment counter of GC by 1
      appropriate child C->insert(P)
  end

node:: iterate (): // Method
  if num points in array > m_leaf // m_leaf=max num points
    create grid from points
    foreach point P in array do
      appropriate child C->insert(P)
    end
  convert this node to inner node
end

```

Listing 1: Inserting points into an MWO.

The points of the nodes are stored in one file per node. Leaf nodes store only point data, while the interior nodes store the point data and the counter for each point in the file.

3.2 Deleting points

Listing 2 shows how points are deleted from an MWO. The points which are going to be deleted have to be selected beforehand, which is done using a Selection Octree [SW11]. All leaf node points inside the Selection Octree are then removed from the MWO, and the hierarchy is checked for consistency.

```

foreach leaf node N inside or intersecting SelOctree do
  define array of deleted points D
  if N is inside SelOctree
    put all points into D
  else
    put only points inside SelOctree into D
  if all points of N are in D
    delete N from disk
  foreach point P in D do
    foreach parent node PN up to root node R do
      decrease counter of grid cell containing P by 1
    root node R->validate()
  end
end

node:: validate (): // Method
  foreach child C that is an inner node do
    C->validate()
  sum up points N in direct and indirect leaf node children
  if N <= m_leaf // m_leaf=max num points
    pull up all points from leaf nodes to this node
    delete empty nodes from disk
    convert this node to leaf node
  end
end

```

Listing 2: Deleting points from an MWO.

Note that there is a subtlety involved when deleting the points. When inserting new points into the hierarchy, the representative points in the grid cells are chosen from the points inserted into a node, e.g., a copy of the first point that falls into a grid cell. When deleting points it might happen that the representative point is the copy of a point that has been deleted, therefore the representative point is not covering the space the remaining points are actually in. This can lead to problems during rendering when using certain types of hierarchy traversals. See Section 4 how these can be solved.

3.3 Rendering

During rendering a depth-first traversal chooses the nodes that will be rendered in the current frame. For this the octree is traversed until the screen-projected size of a node is below a user defined threshold. This node and all of its siblings (inside the view frustum) are then chosen for rendering. Note that the screen-projected sizes of the siblings are not considered, as all

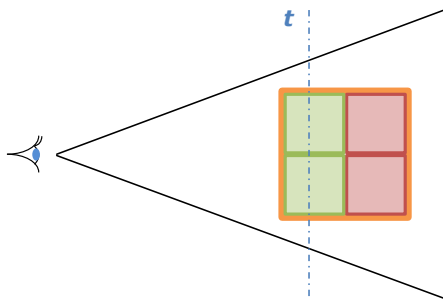


Figure 2: Viewfrustum with an interior node of an MWO and its children. The front of the node has passed the threshold distance t , and the 2 green children have to be rendered. The 2 red children need not be rendered.

siblings in the view frustum have to be chosen to completely replace the parent node (see also Section 4). The loading of the nodes happens asynchronously in a separate thread, so the parent node is still rendered as a coarser LOD level until the children are available.

4 MICHAEL WAND OCTREE OPTIMIZATION

The MWO uses representative points in the interior nodes to enable a LOD mechanism for point clouds. During rendering a depth-first traversal is used for choosing the nodes to render (see Section 3.3). This traversal tends to render more nodes than necessary for the current viewpoint. An example can be seen in Figure 2. The viewpoint is on the left, the orange square represents an interior node of the hierarchy, and the green and red squares represent the interior node's children. The threshold distance t is the distance at which the projected size of the orange node becomes larger than the allowed threshold (for the threshold we use the projected side length of the node's bounding box). A child is thus rendered as soon as its projected double size is larger than the threshold (which is the size of the parent node at the child's position), and its projected size is still smaller than the threshold. In Figure 2 only the front half of the orange node has passed t , therefore only the 2 frontmost children of the node fulfill the conditions to be rendered. The back half of the parent node is still beyond t , so the 2 backmost children need not be rendered, but in a depth-first traversal all children have to be rendered as soon as one child is in front of t .

Instead of a depth-first traversal an importance driven traversal can be used. The importance can be based on different parameters, e.g., the distance to the view point, or how centered the node is on the screen. The importance driven traversal stops when no more nodes can be found which fulfill the conditions to be rendered. In such a traversal no nodes will be rendered whose screen-projected size is too small. In Figure 2, only the 2 frontmost children will be rendered together with

the parent node, as its LOD representation still suffices to appropriately represent the point model in the back area. This often reduces the number of nodes that have to be loaded during rendering, as can be seen in the results (Section 8).

The importance driven traversal requires to render nodes and their children at the same time. Therefore it is important that the points in the interior nodes of the octree actually represent the geometry of the point model, otherwise artifacts as shown in Figure 3 might occur. The center image is rendered with an importance driven traversal, and the points in the interior nodes do not represent the actual geometry of the point model. This can happen if the positions of the points in the interior nodes are not updated after deleting points from the leaf nodes. The image on the right shows no artifacts, and this can be achieved in two different ways, either by using a depth-first traversal, or by using an importance driven traversal and updating the position of the points in the interior nodes during deleting. The pixel overdraw on screen due to the additionally rendered parent nodes for the importance driven traversal is neglectable, as usually less nodes have to be rendered, and nodes that have all children available are not rendered.

This means, depending on the effort spent when deleting points, different rendering traversals can (or have to) be used. Since an importance driven traversal is often favorable, after deleting points we are replacing the points in the interior nodes with points still available in the leaf nodes. For every grid cell of an interior node whose representative point is inside the Selection Octree (and the point's copy in the leaf node has therefore been deleted), we search for a leaf node in the hierarchy that intersects this grid cell, and take a random point (that is also inside the grid cell's bounding box) of this leaf cell as new representative point. This step can be done in a separate traversal, after deleting points from the leaf nodes. The overhead due to this traversal is very low (see results in Section 8).

5 MODIFIABLE NESTED OCTREE RECAP

The MNO is a data structure designed for fast rendering, inserting, and deleting of points in an out-of-core setting [SW11]. This is achieved by a hierarchy of grids which are actually nodes in a region octree [Sam06]. Figure 1 depicts 3 levels of a 2D version of an MNO. Each level further down the hierarchy refines the point cloud representation of the upper levels.

5.1 Inserting points

Listing 3 shows how points are inserted into an MNO. We use a resolution of 2^7 cells on each of the 3 space axis for the grids in the nodes, so up to 128^3 points can be stored at one node.



Figure 3: From the original point cloud (left image) a spherical volume is deleted. The center image shows the artifacts that remain when the interior nodes are not updated after deleting points. In the right image the points in the interior nodes have been updated according to the deleted volume.

```

foreach point P in source point data do // Main loop
  root node R->insert(P)
end

node:: insert (P): // Method
  insert P into grid
  if grid cell GC containing P is empty
    store P in GC
  else
    store P in array of points for child C
    if num points in array for child C >= m_min
      if num points in node without points for C >= m_min
        foreach point PC in array for child C do
          C->insert(PC)
        delete array for child C
      end
    end
  end

```

Listing 3: Inserting points into an MNO.

5.2 Deleting points

```

root node R->delete()
root node R->validate()
end

node:: delete (): // Method
  foreach child C inside or intersecting SelOctree do
    C->delete()
  if node is inside SelOctree
    delete node from disk
  else
    if node is leaf
      delete points inside SelOctree
    else
      foreach point P inside SelOctree do
        intersect grid cell GC containing P with direct and
          indirect child nodes
        delete P
        if at any child node exists a point PC inside GC
          insert PC into GC
      end
    end
  end

node:: validate (): // Method
  foreach child C that is an inner node do
    C->validate()
  foreach child C do
    if num points in C < m_min
      insert all points from C into this node
      delete C from disk
    end
  end

```

Listing 4: Deleting points from an MNO.

Listing 4 shows how points are deleted from an MNO. The points to be deleted are first selected with a Selection Octree [SW11] (see also Section 3). Points are deleted from leaf and interior nodes. Afterwards the hierarchy is checked for consistency.

5.3 Rendering

Rendering of an MNO is done with an importance driven traversal of the hierarchy nodes. Nodes that are in the center of the viewport and close to the viewpoint are most important. Nodes are chosen for rendering until a given maximum number of rendering points is reached, or until further nodes cannot be considered because their screen-projected size is below a pre-defined threshold. Nodes required for rendering but not available in memory are then requested to be loaded from disk. This is done asynchronously, so rendering does not stop while waiting for the points to be loaded.

6 COMPLEXITY ANALYSIS FOR IN-CORE PROCESSING

The MNO [SW11] as well as the MWO [WBB+07] are designed for rendering and accessing large amounts of point data in short time. Search operations on the other hand are not as efficient compared to search operations in specialized data structures. The largest bottleneck is loading points from disk to memory, which can be alleviated to some degree by using an LRU cache. We will first analyze the behavior of the data structures assuming infinite memory to exclude the effects of disk access.

6.1 Building Up Modifiable Nested Octree

In 2D space, building up a Modifiable Nested Quadtree is similar to building up a point quadtree [FB74], as both data structures store points in interior nodes as well as leaf nodes. While the point quadtree subdivides space at given data points, the Modifiable Nested Quadtree creates a hierarchy of congruent squares and each hierarchy level subdivides the space at the center of the previous hierarchy level. A point quadtree stores at each node exactly one point. The average build up

Hierarchy	Build up		Delete	
	Time	Space	Leaf Node	Inner Node
MNO	$O(N)$ to $O(N \log(N))$	$O(N)$	$O(\log(N))$	$O(\log(N))$
MWO	$O(N)$ to $O(N \log(N))$	$O(NC)$	$O(M + \log(N))$	$O(M + \log(N))$

Table 1: The time and space complexities for building MNO and MWO hierarchies, and for deleting points from leaf and inner nodes.

Hierarchy	Search		
	Single Point	Sphere	Cuboid
MNO	$O(\log(N))$	$O(F + 2^n M)$	$O(F + 4^n M)$
MWO	$O(M + \log(N))$	$O(F + 2^n M)$	$O(F + 4^n M)$

Table 2: The time complexities for searching points in MNO and MWO hierarchies.

Hierarchy	Build up Time		Delete	
	Worst Case	Average	Single Point	Average
MNO	$O(CN + CN \log(N))$	$O(kC + N \log(N))$	$O(C + C \log(N))$	$O(kC + N \log(N))$
MWO	$O(CN + CN \log(N))$	$O(kC + N \log(N))$	$O(C + M + C \log(N))$	$O(kC + NM + N \log(N))$

Table 3: The time complexities for the out-of-core build up and deleting points from MNO and MWO hierarchies.

Hierarchy	Search		
	Single Point	Sphere	Sphere Avg.
MNO	$O(C \log(N))$	$O(CF + 2^n(M + C))$	$O(kC + F + 2^n M)$
MWO	$O(M + C \log(N))$	$O(CF + 2^n(M + C))$	$O(kC + F + 2^n M)$

Table 4: The time complexities for searching points in MNO and MWO hierarchies.

time for a point quadtree is proportional to $N \log_4(N)$ where N is the total number of points used for building the quadtree [FB74].

In 3D it becomes a point octree. For a complete point octree the average build up time becomes roughly $N \log_8(N)$. This estimation can be derived by calculating the total path length (TPL) [WN73] for a completely filled point octree. The TPL is defined to be the summed up path length when searching all nodes in a tree, and starting the search always at the root node. The path length of a node and one of its direct children is 1.

Like a point octree an MNO stores points at interior nodes as well as leaf nodes. In an MNO though up to m (e.g., 128^3) points can be stored at each node. When using a hash table for storing the points, inserting a point into a node takes constant time on the average [CLRS09]. The possible reduction is therefore dependent on the ratio m/N . If this ratio is ≥ 1 , then complexity becomes $O(N)$. For ratios approaching 0, the complexity again approaches $O(N \log_8(N))$, which is equivalent to $O(N \log(N))$ (see Table 1).

For point clouds from laser scanners the number of points that are stored at each node is much smaller than m , since the points are distributed highly nonuniform in space. Therefore $m \gg N/k$, where k is the number of nodes in an MNO. Huge point clouds where $N \gg N/k$ show a complexity of roughly $O(N \log(N))$, while smaller point clouds tend to show a $O(N)$ complexity. The space requirements for storing an MNO on disk are $O(N)$, since no additional points have to be saved.

6.2 Building Up Michael Wand Octree

In 2D space a Michael Wand Quadtree can be compared to a bucket point region quadtree (bucket PR quadtree) [Sam06]. This is a quadtree where the children of a node subdivide the node's space into four congruent squares. The data points are only stored in leaf nodes, and the interior nodes are used to direct the traversal of the quadtree. Each leaf node stores up to m_{leaf} points. The Michael Wand Quadtree stores for the level-of-detail mechanism (see Section 3) additionally up to $m_{interior}$ points in the interior nodes.

In an octree where the points are only stored at the leaf nodes, the time complexity for build up is $O(N \log_8(N)) \Rightarrow O(N \log(N))$, according to the total path length (see also Section 6.1). Using buckets at interior nodes and leaf nodes the complexity for building up an MWO changes depending on the ratio m_{leaf}/N . For ratios ≥ 1 the time complexity becomes $O(N)$, and for ratios approaching 0 the time complexity approaches $O(N \log(N))$ (see Table 1).

The distribution of the samples in the point cloud determines the fan out of the hierarchy nodes as well as the fill rate of the grids at the interior nodes. If the point cloud represents a line, i.e., an object with dimensionality $d = 1$, the fan out of the interior nodes will be similar to a binary tree, i.e., each node will have about 2 child nodes. Similar for $d = 2$ or $d = 3$ the interior nodes will have about 4 or 8 children. Assuming a grid with G cells per axis, then the grid has approximately G^d cells filled. Note that since the MWO uses point buckets at the nodes, their fan out will rather represent the global characteristic of the data set. The granularity

of the nodes is too big to capture the true dimensionality. For a better estimation of the dimensionality, the points within the nodes have to be used.

We are not making any assumptions about the point data, therefore an initial estimate for d , without having previously built MWOs of similar data sets, is very difficult. Instead we use the average number of points in the interior nodes, X , of a previously built MWO as parameter for the approximation of the dimensionality. With X and G given, then $d = \log_G(X)$. From this an estimation for the fan out F can be derived as $F = 2^d$.

When given the total number of points N in a data set and having an estimation for F and the related dimensionality d , the space requirements for the interior nodes of a new MWO of a similar data set can be evaluated using the following 3 equations.

The total number of leaf nodes L can be estimated by using F as an approximation to the number of new leaf nodes that are created when a leaf node has to be split during build up. The maximum number of leaf nodes is then bound by

$$L = \frac{N * F}{m_{leaf}} \quad (1)$$

and on the average each of the new leaf nodes will hold at least m_{leaf}/F points. Having an estimation for L , the total number of interior nodes I_{nodes} can be estimated with

$$I_{nodes} = \frac{L - 1}{F - 1} \quad (2)$$

which is the number of interior nodes for any tree with an average fan out F . By using the dimensionality d together with G , the number of grid cells on one axis, the total number of points in the interior nodes I_{points} can be evaluated by

$$I_{points} = I_{nodes} * G^d \quad (3)$$

The space complexity can be written as $O(NC)$, where C is a constant depending on the estimated parameters d and F as well as on the given parameters G and m_{leaf} . For $L \gg 1$ the term $L - 1$ can be replaced by L and Equation (2) can thus be approximated by $L/(F - 1)$. With this simplification C can be written as

$$C = 1 + \frac{F * G^d}{(F - 1) * m_{leaf}} \quad (4)$$

For example a 2D object with $d = 2$ and $F = 4$ has a value of $C = 1 + (4/3) * (G^2/m_{leaf})$.

When using the parameters suggested by [WBB+07], $G = 128$ and $m_{leaf} = 100.000$, then $C = 1.218$ for a

two-dimensional object and $C = 24.97$ for a complete 3D volume. This means that for point clouds representing a single 2D area the number of additional points I_{points} is estimated to be 21% of the number of original points, while for a point cloud representing a complete 3D volume it is 24 times the number of original points. For the point clouds we tested with these parameters the number of additional points was 40% of the original points, suggesting the dimensionality of the data sets is almost totally 2D.

6.3 Searching

Searching for a single point in one of the two hierarchies means first searching for the node that holds the point and then searching for the point within this node. The complexities are listed in Table 2. The search for a point within an MNO node is of constant complexity $O(1)$, since the points in the nodes are stored inside grids (which are built once with an $O(N)$ complexity). For an MWO the search for a point inside the leaf node is of $O(M)$, if no search data structure (like a kd-tree [Ben75]) for the points in the leaf nodes has been created.

6.3.1 Modifiable Nested Octree

A range search in an MNO is similar to the range search in a PR quadtree, only in 3 dimensions. A PR quadtree subdivides the space hierarchically into congruent squares. All points are stored in its leaf nodes, one point per leaf node. A range search in a PR quadtree, where the range is a rectangular area limited by axis aligned lines, has a complexity of $O(F + 2^n)$, where F is the number of reported points and n is the maximum depth of the PR quadtree [Sam06]. The 2^n term is derived from the maximum number of nodes that one line of the rectangle can intersect with. A PR quadtree in the 3D case becomes a PR octree. Searching all points in a PR octree that are inside an axis aligned rectangular cuboid is proportional to $O(F + 4^n)$, where 4^n accounts for the lateral surfaces of the cuboid intersecting the octree nodes.

A range search in an MNO has to take interior nodes as well as leaf nodes into account. This means all levels of the MNO contribute to the search algorithm complexity. This results in a performance proportional to $O(F + \sum_{i=0}^n 4^i) \Rightarrow O(F + 4^n)$. The complexity does therefore not increase for an MNO.

When using a sphere as the search range, it has to be discretized for counting the nodes intersecting the sphere. The number of nodes intersecting a discretized sphere is the solution to the diophantine inequation $(R - 1/2)^2 \leq x^2 + y^2 + z^2 < (R + 1/2)^2$ [And94], where $R \in \mathbb{N}$ is the radius of the discrete sphere, and $x, y, z \in \mathbb{Z}$. For increasing R it converges to $4\pi R^2$. The worst case complexity of the range search is found when using

$R = 2^n/2$. This is the radius of the inscribing sphere for the axis aligned bounding box of the root node, expressed in number of nodes at depth n . The complexity of this range search is thus proportional to $O(F + \sum_{i=0}^n 4\pi 2^{2i}/2^2) \Rightarrow O(F + \sum_{i=0}^n 2^{2i}\pi) \Rightarrow O(F + 2^n)$.

After identifying the nodes intersecting with the search range, all points within the intersecting nodes have to be tested if they are inside or outside the search range. This is of linear complexity $O(M)$ per node, where M is the maximum number of points in a node. In total the range search complexity for a cuboid inside an MNO is $O(F + 4^n M)$ and for a sphere it is $O(F + 2^n M)$.

6.3.2 Michael Wand Octree

During a range search in an MWO points only have to be searched for in the leaf nodes, contrary to an MNO, but since the upper levels of the MNO do not contribute to the search complexity, both hierarchies have the same complexities (see Table 2).

6.4 Deleting

Deleting a point from a leaf node means basically searching for it, and on the average this has a time complexity of $O(\log(N))$ (see Table 2). In an MWO points are always deleted from the leaf nodes and the interior nodes are updated later. In an MNO points are directly deleted from leaf nodes and interior nodes.

6.4.1 Modifiable Nested Octree

In a node the search for the grid cell a point falls into can be done in $O(1)$. When a point is deleted from a leaf node, the LOD structure remains intact. When a point is deleted from an interior node, two more steps are required, i.e., finding a replacement point and pulling it up to the node where the other point has been deleted from.

A replacement point is searched for in a leaf node that encompasses (or is inside of) the bounds of the node's grid cell the other point was deleted from ($O(\log(N))$ complexity). This replacement point is then deleted from the leaf node and inserted into the node the other point was deleted from ($O(1)$ complexity). Therefore the total complexity of deleting a point from an interior node is $O(2\log(N)) \Rightarrow O(\log(N))$.

6.4.2 Michael Wand Octree

The search for a point in a leaf node is done in linear time $O(M)$, where $M = m_{leaf}$. After deleting the point from the leaf node, the interior nodes on the path from the leaf node to the root node have to update the counters of the grid cells into which the point falls ($O(\log(N))$ complexity). The total complexity for deleting a point is $O(M + 2\log(N))$.

When using the optimization of Section 4 the points of the interior nodes inside the Selection Octree have to be

updated. They have to look for a replacement point in an existing leaf node. With this additional step the total complexity for deleting a point is $O(M + 3\log(N)) \Rightarrow O(M + \log(N))$.

7 COMPLEXITY ANALYSIS FOR OUT-OF-CORE PROCESSING

Having only a limited amount of main memory, the point data can become larger than the available main memory and an out-of-core management for the data becomes necessary. This way the points of the hierarchy nodes are streamed from disk during processing. Accessing the disk has a large overhead compared to memory access, therefore reducing the number of disk accesses is important for the efficiency of the out-of-core data management. In the worst case, e.g., if the locality of an operation on the point cloud is bad, each access to a hierarchy node is preceded by a disk access.

One strategy to reduce the number of disk accesses is to use an LRU cache, which manages nodes according to the time they were last used. If points for a new node have to be loaded from disk, the points of the node which has been accessed the longest time ago are swapped out of memory. Generally, using a memory caching method can increase performance a lot. The parameters influencing the efficiency of a LRU cache are the size of the cache and the access pattern of the operations to the nodes of the point-cloud hierarchy.

7.1 Building up and Inserting

The complexities for building up a point-cloud hierarchy are given in Table 3. C is a huge constant representing disk access. The term NC accounts for writing every point to disk, and $CN\log(N)$ for loading the necessary nodes. In the average case, when using an LRU cache, nodes can often be accessed in the cache. Parameter k is the total number of nodes in the hierarchy. To reduce disk access time a solid state drive (SSD) can be used.

7.2 Searching

The complexities for searching are given in Table 4. We give only the complexities for the spherical search, as the cuboid search can be derived analogously. In the average case the parameter kC accounts for loading and reporting the found points.

7.3 Deleting

The complexities for deleting are given in Table 3. C accounts for writing the changed nodes to disk. In the average case parameter kC accounts for loading and saving the nodes. The dominating terms are dependent on the distribution of the deleted points. If only few points per node are deleted, kC will be the dominating term, otherwise $N\log(N)$ will be dominating.

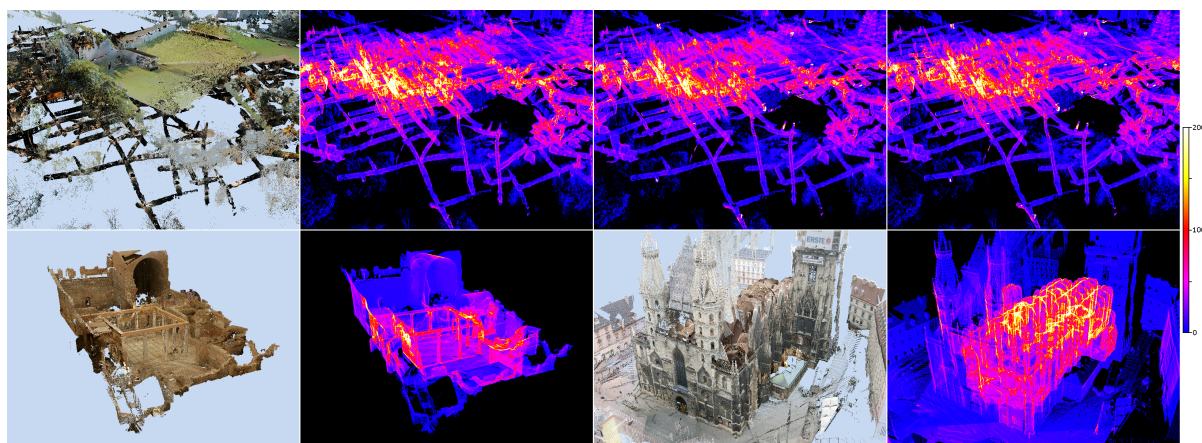


Figure 4: In the top row the Domitilla model together with some overdraw heat-map visualizations is shown, from 2nd image left to right this are the MNO, the MWO with 30k leaf nodes and priority-based traversal, and the MWO with 30k leaf nodes and depth first traversal. In the bottom row the Hanghaus and Stephansdom models are shown, and the MNO is used for heat-map visualization. All traversals use a maximum projected grid cell size of 1 pixel and are rendered with 1 pixel per point.

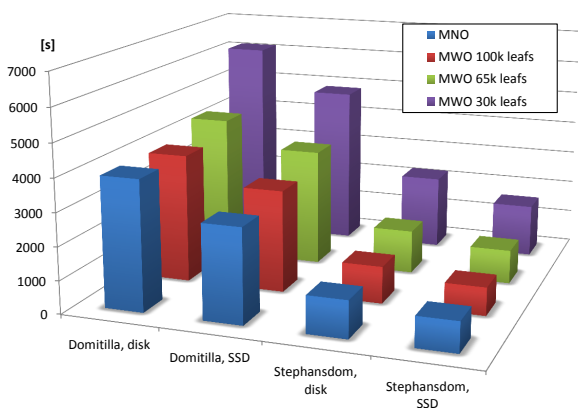


Figure 5: Buildup timings for the Domitilla and Stephansdom models on disk and on SSD.

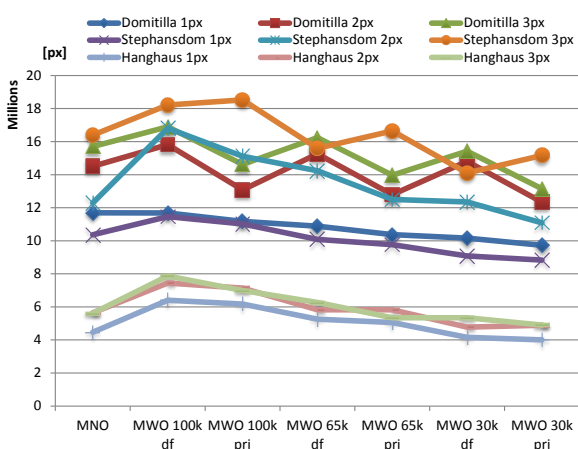


Figure 6: Total number of pixels drawn at different settings for the projected grid cell size.

8 RESULTS

We implemented the MNO and the MWO data structures in our point rendering and editing framework, to be able to compare both data structures directly. Our test computer has an Intel Core i7-2600 quad core CPU, 16 GB RAM, two 10.000 rpm hard disks in a RAID 0, and 4 solid state drives (SSDs) in a RAID 0. All operations with the data structures use an LRU cache of 100 million points. Rendering is done in OpenGL.

For the tests we use 3 point data sets, the Domitilla catacomb with 1.92×10^9 points (courtesy of the FWF START project "The Domitilla-Catacomb in Rome."), the Stephansdom with 460×10^6 points, and the Hanghaus with 14.7×10^6 points. In Figure 4 the data sets (together with some overdraw heat maps) are shown. From each data set we build 4 models, one MNO with minimum 1000 points per node, and 3 MWOs with 30k, 65k, and 100k maximum points per leaf node. The space requirements for an MWO increase with smaller leaf node size, e.g., for the Domitilla model the MWO with 30k leaf nodes is about 2.2 times larger than the original data. This leads to higher build up times as can be seen in Figure 5. Build up times for all Hanghaus models are 30 seconds or less. Furthermore we measured the total number of fragments written to the 800x600 pixels viewport in 21 different settings per model. For each model we tested a maximum projected grid cell size of 1, 2, and 3 pixels, and the models were then rendered with screen aligned OpenGL points of the according size. For each grid cell size we rendered the MNO model and the 3 MWO variants. Each MWO was rendered once with a priority-based traversal, and once with a depth first traversal. All renderings were done from one viewpoint per model. The results are given in Figure 6, and except for the Stephansdom

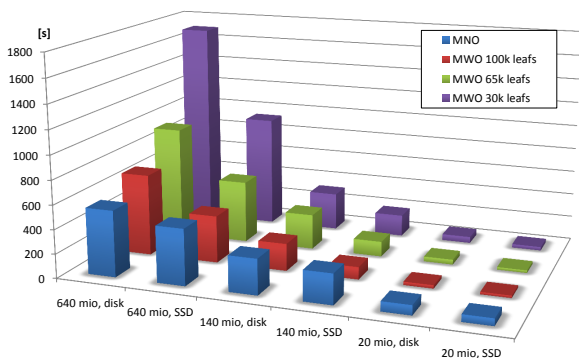


Figure 7: Timings for deleting differently sized selections from the Domitilla model.

MWOs rendered with 3 pixels and the Hanghaus 30k MWO rendered with 2 pixels projected grid size, the priority-based traversal causes less overdraw. Editing operations were tested by deleting different sized selections from the largest point cloud, the Domitilla model. The results in Figure 7 show that the usage of an SSD and also a smaller selection size do not speed up the operation for the MNO as for the MWO models. This is caused by a more expensive pull up operation in the MNO, which uses nodes in the LRU cache. The time overhead for updating the interior nodes after deleting is about 1% for the largest, 4% for the medium, and about 10% for the smallest selection.

9 CONCLUSION

We have presented a complexity analysis for the MNO and the MWO, two data structures which can be used for editing and rendering huge point clouds. The data structures show similar complexities for build up and editing operations. The memory requirements and thus processing times for the MWO are dependent on the leaf node size, while the memory requirements for the MNO are the same as for the original data set. The MNO is better suited for large editing operations, as less nodes have to be processed. The MWO (especially with 30k leaf nodes) is better suited for rendering, as often less points are required to render a point cloud from a certain viewpoint, resulting in higher frame rates. We also introduced a lightweight extension to the points deleting algorithm for the MWO, which allows to use efficient node selection strategies during rendering, leading to less pixel overdraw.

10 REFERENCES

- [And94] Andres, E. Discrete circles, rings, and spheres. *Computers and Graphics*, 18(5):695-706, 1994.
- [Bay08] Bayer, R. B-tree and UB-tree. *Scholarpedia*, 3(11):7742, 2008.
- [Ben75] Bentley, J.L. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9):509-517, September 1975.
- [BM72] Bayer, R., and McCreight, E.M. Organization and Maintenance of Large Ordered Indexes. *Acta Informatica*, 1:173-189, 1972.
- [CLRS09] Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C. *Introduction to Algorithms*, Third Edition. The MIT Press, 3rd edition, 2009.
- [Den68] Denning, P.J. The Working Set Model for Program Behavior. *Communications of the ACM (CACM)*, 11(5):323-333, May 1968.
- [FAR13] FARO Technologies, Inc. <http://www.faro.com/>, 2013.
- [FB74] Finkel, R.A., and Bentley, J.L. Quad Trees A Data Structure for Retrieval on Composite Keys. *Acta Informatica*, 4:1-9, 1974.
- [GM04] Gobbetti, E., and Marton, F. Layered point clouds. *Computers & Graphics*, 28(6):815-826, 2004.
- [Mic03] Microsoft. How NTFS Works. Microsoft TechNet Webpage, 2003.
- [PSL05] Pajarola, R., Sainz, M., and Lario, R. Xsplat: External memory multiresolution point visualization. In *Proceedings IASTED International Conference on Visualization, Imaging and Image Processing*, pages 628-633, 2005.
- [Rie13] Riegl Laser Measurement Systems. <http://www.riegl.com/>, 2013.
- [Sam06] Samet, H. *Foundations of Multidimensional and Metric Data Structures*. Elsevier, 2006.
- [SW11] Scheiblauer, C., and Wimmer, M. Out-of-Core Selection and Editing of Huge Point Clouds. *Computers & Graphics*, 35(2):342-351, April 2011.
- [Vit08] Vitter, J.S. Algorithms and Data Structures for External Memory. *Foundations and Trends in Theoretical Computer Science*, 2(4):305-474, 2008.
- [WBB+07] Wand, M., Berner, A., Bokeloh, M., Fleck, A., Hoffmann, M., Jenke, P., Maier, B., Staneker, D., and Schilling, A. Interactive Editing of Large Point Clouds. In *Symposium on Point Based Graphics*, pages 37-45, Prague, Czech Republic, 2007. Eurographics Association.
- [WBB+09] Wand, M., Berner, A., Bokeloh, M., Fleck, A., Hoffmann, M., Jenke, P., Maier, B., Staneker, D., and Parys, R. Xgrt extensible graphics toolkit, 2009.
- [WN73] Wong, C.K., and Nievergelt, J. Upper Bounds for the Total Path Length of Binary Trees. *J. ACM*, 20(1):1-6, January 1973.

A segmentation method for nuclei identification from sagittal images of *Drosophila melanogaster* embryos

Daniela Justiniano de Sousa^a, Maira Arruda Cardoso^b, Paulo Mascarello Bisch^b,
Francisco José Pereira Lopes^{b,c,d}, Bruno Augusto Nassif Travençolo^a

^aFaculdade de Computação, Universidade Federal de Uberlândia,
38400-902, Uberlândia, MG, Brazil

^bInstituto de Biofísica Carlos Chagas Filho, Universidade Federal do Rio de Janeiro,
21941-902, Rio de Janeiro, RJ, Brazil

^cUniversidade Federal do Rio de Janeiro, Polo de Xerém,
25245-390, Duque de Caxias, RJ, Brazil

^dInmetro - National Institute of Metrology, Quality and Technology, Brazil

daniela@mestrado.ufu.br, mairaarrudacardoso@gmail.com, pmbisch@biof.ufrj.br,
flopes@ufrj.br, travencolo@gmail.com

ABSTRACT

This paper proposes a segmentation method for sagittal images obtained from *Drosophila melanogaster* embryos at early stages of development. The proposed method operates in the spatial domain and uses traditional filters and segmentation techniques for image processing. However, one common problem encountered after the segmentation of sagittal images is the merged nuclei. In order to split these nuclei, our approach uses the curvature of the shape of the merged nuclei to find the regions of division between the nuclei. The proposed method was compared to other techniques and it achieved better results. In general, the obtained results have shown good performance and were able to identify individual nuclei, providing an efficient and accurate solution for segmentation of nuclei in images obtained from the sagittal plane of *Drosophila* embryos.

Keywords

Image segmentation, Curvature, Nuclei segmentation, *Drosophila melanogaster*.

1 INTRODUCTION

The segmentation is an important area of study for image analysis and computational vision. The term image segmentation refers to the partition of an image into a set of regions that cover it, aiming identifying or extracting regions of interest, in other words, all relevant semantic content to the application. Regions of interest should be uniform and homogeneous with respect to some characteristic, such as gray level, color, or texture. The segmentation step is usually considered critical, since, through it is possible to identify, recognize and classify digital objects. In general, it represents one of the most important steps, considering that its result may to determine the success or failure of computerized analysis procedures [Shap01, Gonz08].

Image segmentation performs a crucial role in bioimage informatics [Peng08, Zhan12]. In some cases, the complexity or particular characteristics of biological data hampers the process of segmentation, thus leading to the development of a wide range of segmentation methods addressing specific problems in biological applications. Normally, such methods make use of prior knowledge for the particular objects of interest and other possible structures in the image. In this context, we propose a new segmentation method for application to nuclei segmentation from sagittal images obtained from early *Drosophila melanogaster* embryos (e.g., Fig. 1(a-d)). The proposed method is based on analysis of curvature of the shape of the nuclei and operates in the spatial domain. In this case, aiming to achieve the desired segmentation it uses the knowledge about the geometric characteristics of objects contained in this type of image.

The *Drosophila melanogaster*, commonly known as the “fruit fly”, is an important model in biological research, particularly in the study of gene expression regulatory networks, spatial expression pattern and cellular differentiation [Crow12, Gilb03]. Recent advances in molecular biology and microscopy techniques sub-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

stantially increased the amount of images generated for investigations on *Drosophila* development, especially images of gene expression patterns. In order to extract information from these images, several computational methods has been proposed [Pous04, Pan06, Jans05, Rbel06, Pisa09, Kozl08], followed by an increasing number of databases of images available on Internet as, for example, Flyex [Pisa09], BDGP [Toma07] and BDTNP [Fowl08]. A common feature observed in the images in those databases is that the data (anatomical or gene expression patterns) are obtained from the surface of the embryo (or 3D in the case of BDTNP). None of these works deals with images obtained from sagittal planes passing through the embryo.

Sagittal images currently gained great interest to *Drosophila* community. By allowing the visualization of cells and tissues, localization of genes and proteins, as well the visualization of the dynamics of gene expression in embryos *in vivo* [Greg07], this type of image can unveil new gene interactions and new findings about the relationship between anatomy and gene expression during development. Therefore, the image segmentation method proposed in this paper refers to sagittal images of early *Drosophila* embryos. More specifically, we are interesting in the identification of nuclei of the embryos, i.e., to define nuclei masks. This type of analysis increases the amount of features that can be extract from the images, as the obtained masks can be used to characterize several morphological properties, as well serve of basis to characterize gene expression patterns. For example, the nuclear mask can be used to quantify data on gene expression in each embryo nucleus, or to characterize the amount of gene products inside the embryo nuclei [Pisa09].

In order to obtain the nuclear mask, the proposed method add a post-processing step on the result of Otsu segmentation method [Otsu79]. This extra step is necessary because the result of initial segmentation results in many blocks of merged nuclei, which should be separated in order to obtain a correct identification of the nuclei. For separation of merged nuclei, we propose an analysis of the curvature of the shape of the merged blocks. In order to evaluate the efficiency of the method, it was performed several experiments with real images, which were collected by using confocal microscopy. The obtained results show efficiency and good performance in the identification of cell nuclei contained in the images processed.

2 PREVIOUS WORKS

Most of the works involving *Drosophila* embryos refer to the surface of the embryos [Pous04, Jans05, Pisa09, Surk08, Jaeg04] and the nuclear mask is usually obtained through of technique proposed by Kosman [Kosm99]. More sophisticate

techniques deals with 3D data [Huan08, Luen06]. However, the content of both 3D data and surface images are different from sagittal images.

In addition, to the best of our knowledge, it has not been proposed any works related to the sagittal image of *Drosophila* embryos. Few works deal with computational methods in order to extract relevant information from sagittal images. Houchmandzadeh et al. [Houc02] used a sliding rectangle of fixed size perpendicular to the edges of the embryo while probing the average pixel intensity under the rectangle, while Gregor et al. [Greg07] used the same approach, but a circular mask were used instead of a rectangle. However, none of these approaches can identify individual nuclei.

Other works not directly related to *Drosophila* image could be used for sagittal images, as the works by Costa et al. [Ferr97], Malpica et al. [Malp97] and Bala [Bala12]. In this paper we analysed the results of these three methods, along with the method of Kosman [Kosm99], in order to compare to the results of our method. All of these proposals address variations of watershed segmentation algorithm [Vinc91, Beuc79], which is considered a powerful technique to isolate and recognize cell nuclei in biological images [Chen12, Chan12].

3 METHODOLOGY

Image segmentation consists in the extraction and identification of regions of interest contained in a image [Gonz08]. In the case of sagittal images of early *Drosophila melanogaster* embryo, the regions of interest are those concerning to the cell nuclei. In this case, the objective of the segmentation is to obtain a nuclear mask image, i.e., a binary image where the pixels with value equal to 1 represent cell nuclei and the pixels with a value 0 represents the background of the image [Pisa09]. For each embryo, a binary nuclear mask must to be constructed. This mask shows where the individual nuclei of an embryo are located [Pous04, Pisa03].

This section describes the proposed segmentation method to obtain the nuclear mask image. The method consists of three main steps: pre-processing, image binarization and a post-processing step to split merged nuclei.

3.1 Pre-processing

Sagittal images from *Drosophila* embryos were obtained using confocal microscopy. These images present some degree of noise, which should be attenuated in order to optimize the results of subsequent operations, such as image binarization. In the pre-processing stage, a simple spatial filtering can

be considered for noise removal. In addition, it is interesting to include some sharpen operations in order to highlight the edges of the nuclei presented in the image.

In the images analyzed in this work the noise are located on the extremities of cell nuclei, characterized by the random occurrence of lightness values significantly different from pixel values referring to the foreground of image. Gaussian smoothing was applied in order to reduce these noise [Gonz08].

With regard to highlighting techniques, two operations are proposed, the linear intensity transformation and the morphological highlighting [Solo10, Gonz08]. The linear highlighting was applied in order to emphasize and improve the appearance of images. It is the simplest way to modify contrast and the brightness of a image, where the mapping function is represented by equation $r = ax + b$, such that x is the original pixel value and r is the pixel value after the transformation. The parameter a (angular coefficient) affects jointly the contrast and brightness of the resulting image and b (linear coefficient) adjusts only the brightness [Solo10]. After this linear mapping function, it was applied in the images the morphological highlighting [Shih09, Soil99], expressed by Eq. (1).

$$I(x,y) = (G(x,y) + T_{hat}(G)) - B_{hat}(G), \quad (1)$$

where $I(x,y)$ represents the sharpen image, $G(x,y)$ the smoothed input image and $T_{hat}(G)$ and $B_{hat}(G)$ the result of the application of “top-hat” and “bottom-hat” operators, respectively.

The technique of morphological highlighting (Eq. (1)), constructed from the combination of the operations “top-hat” and “bottom-hat” [Shih09], aims to increase the visual discrimination between foreground and background of processed images, through addition of the “top-hat” result to the smoothed image, followed by the subtraction of “bottom-hat” image. The operations involved in morphological highlighting include binary opening and closing. The binary opening consists in applying in the binary image the operation of erosion followed by a dilation, while the binary closing consists of a dilatation followed by an erosion.

The operation “top-hat” detects the peaks (local maxima) of smoothed image $G(x,y)$, and is defined by the difference between $G(x,y)$ and the prior result of its morphological opening [Shih09], as defined in Eq. (2). The symbol SE defines the structuring element, and it is usually chosen according to the characteristics of the image.

$$T_{hat}(G) = G(x,y) - (G(x,y) \circ SE). \quad (2)$$

On the other hand, the operation “bottom-hat” detects the valleys (local minima) of the smoothed image

$G(x,y)$, and is defined by difference between the result of its morphological closure [Shih09] and the image $G(x,y)$, as is expressed by Eq. (3).

$$B_{hat}(G) = (G(x,y) \bullet SE) - G(x,y). \quad (3)$$

The application of morphological highlighting emphasizes the pixels belonging to cell nuclei of sagittal images, while it attenuates the intensity values of the pixels between the nuclei.

3.2 Image Binarization

In order to binarize the pre-processed images it was verified the performance of traditional binarization techniques existing in the literature, as thresholding, edge detection and region growing [Gonz08]. It was verified that, when applied separately, such techniques are insufficient for obtaining a precise nuclear mask [Huan08]. These techniques were not able to provide full separation of the nuclei (i.e., they have many merged nuclei, as illustrated in Fig. 1(e-h)). This factor is influenced by morphological features (the nuclei may indeed be merged) and by small variations of light intensity existing between the nuclei.

After several tests, the Otsu thresholding [Otsu79] was chosen for image binarization. This is an unsupervised segmentation method which relies on discriminant analysis and searches for an optimal threshold T^* which maximizes the variance between classes C_0 and C_1 , which, in turn, represent the objects – the nuclei – and the background image (or vice versa) [Solo10, Gonz08]. Many approaches for cell nuclei detection in fluorescence images rely of the Otsu segmentation method [Xion06b, Quel10, Xion06a, Yan08]. In addition, in order compensate the non-uniformity of the lightning in some images, the images were subdivided into rectangles and, for each subimage, the Otsu method was applied individually. This subdivision considerable improved the results of the binarization.

However, as illustrated in Fig. 1(e-h), the result of Otsu binarization also resulted in an image with a significant presence of two or more nuclei unified. These nuclei are identified in yellow color in Fig. 1(e-h). So, after binarization it is necessary an additional step to split the merged nuclei.

3.3 Splitting merged nuclei using curvature

After the binarization of the image using Otsu method, there were still several merged nuclei. In order to split these nuclei, we propose the use of the curvature of the shape of the merged regions. The curvature quantifies the variations in the contour of a shape. It is very useful in defining of relevant geometrical properties, such

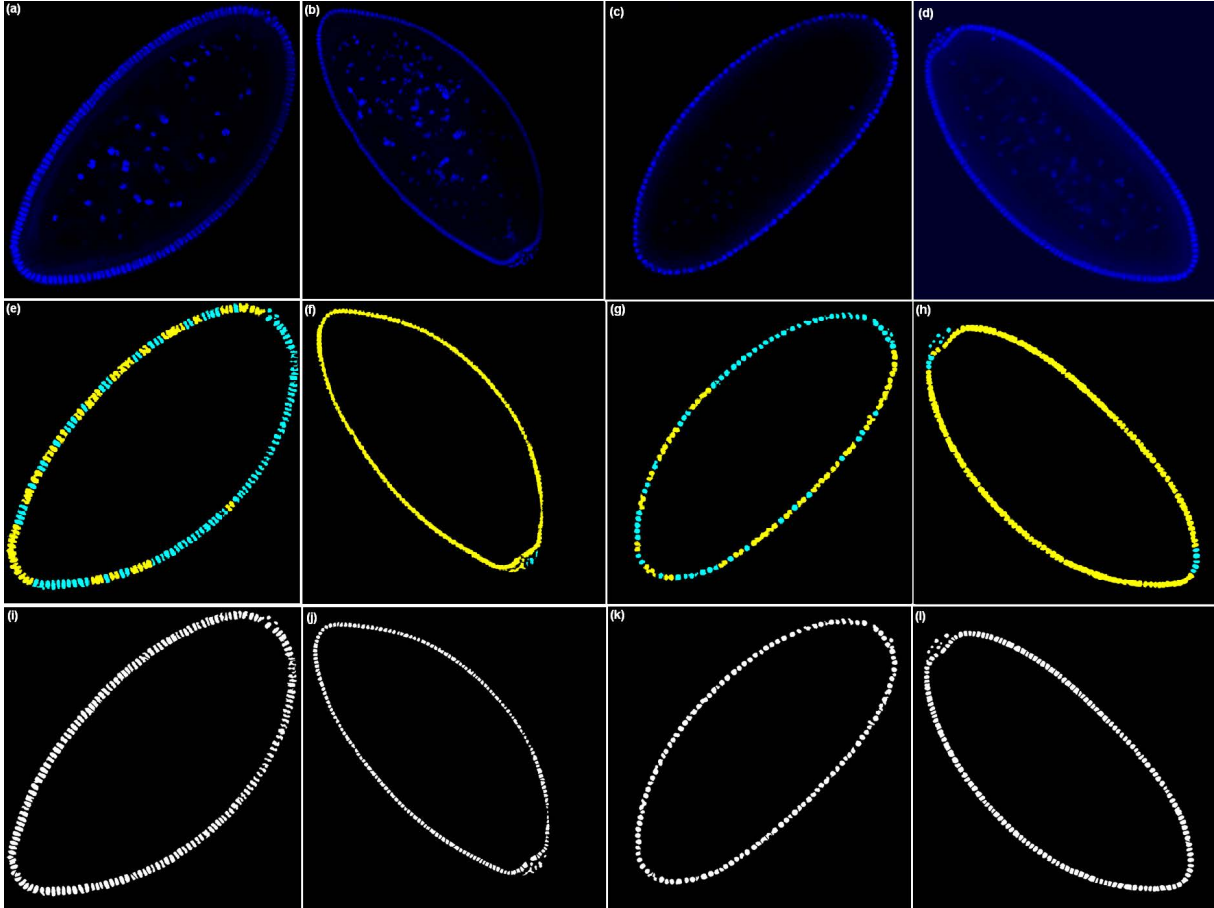


Figure 1: Original and segmented images. (a-d) Original images, obtained by confocal microscopy at the sagittal plane of *Drosophila melanogaster* embryos. (e-h) Results after of Otsu binarization. Merged nuclei are identified in yellow while rightly detected nuclei are shown in cyan. It can be seen that several nuclei remain unified after the binarization. The components outside the border of the embryo (e.g. cells inside the embryo) were removed manually after the binarization. (i-l) Results obtained after the application of the proposed methodology. Note that the merged nuclei are now separated.

as corners, valleys, concave regions and convex, and straight lines [Attn54, Levi85].

In digital images, the curvature of an object can be estimated from its parametric contour $C(t)$ (Eq. (4)), i.e., the representation in which the object contour is described in terms of a single parameter t , where $t = 1, \dots, n$ is the index of the positions of the sequential pixels $(x(t), y(t))$ defining the contour of the object and n is the number of pixels of the contour of the object.

$$C(t) = [x(t_1), y(t_1)], [x(t_2), y(t_2)], \dots, [x(t_n), y(t_n)]. \quad (4)$$

The curvature k for each element of a discrete contour $C(t)$ can be obtained according to Eq. (5), where $x'(t)$ and $y'(t)$ are the first order derivative with respect to x and y , and $x''(t)$ and $y''(t)$ are the second order derivative in relation to x and y , respectively [Cost09].

$$k(t) = \frac{x'(t)y''(t) - y'(t)x''(t)}{(x'(t)^2 + y'(t)^2)^{\frac{3}{2}}}. \quad (5)$$

In this work the discrete derivatives necessary for curvature estimation was obtained using the derivative property of the Fourier Transform (\mathfrak{F}) [Cost09]. Let $X(f)$ and $Y(f)$ be the Fourier Transform of the signals $x(t)$ and $y(t)$ respectively, the first and second derivatives of these signals are defined by Eqs. 6a-6d.

$$x'(t) = \mathfrak{F}^{-1}(i2\pi fX(f)), \quad (6a)$$

$$x''(t) = \mathfrak{F}^{-1}(-2\pi f)^2 X(f), \quad (6b)$$

$$y'(t) = \mathfrak{F}^{-1}(i2\pi fY(f)), \quad (6c)$$

$$y''(t) = \mathfrak{F}^{-1}(-2\pi f)^2 Y(f). \quad (6d)$$

In the case of discrete functions, the Eqs. 6a-6d normally generates noisy derivatives. The solution in this case is to apply a Gaussian filter in the function by including in the Eqs. 6a-6d the term $G(f, \sigma) = \exp(-2(\sigma\pi f)^2)$, a Fourier Transform of a Gaussian function with zero mean and standard deviation σ , defined in the frequency space. Thus,

the effect of this filter is smoothing, where it reduces (or even eliminated) the influence of noise and small undesired details in the contour of the analyzed shape.

$$x'(s) = \mathfrak{F}^{-1}(i2\pi fX(f)G(f, \sigma)), \quad (7a)$$

$$x''(s) = \mathfrak{F}^{-1}(-(2\pi f)^2 X(f)G(f, \sigma)), \quad (7b)$$

$$y'(s) = \mathfrak{F}^{-1}(i2\pi fY(f)G(f, \sigma)), \quad (7c)$$

$$y''(s) = \mathfrak{F}^{-1}(-(2\pi f)^2 Y(f)G(f, \sigma)). \quad (7d)$$

The estimation of smoothed curvature is calculated for all connected components with two or more nuclei unified. These components are identified by their size, been analyzed the components that are larger than a preestablished value. This value is defined based on the average size of the nuclei. The main idea in using the curvature is to find points in the image where we can trace a straight line separating the merged nuclei. The same idea was employed in the work of Beletti et al. [Bele05], which used the curvature in the analysis of sperm images in order to separate head from the tail. The process of using the curvature can be better understood by analyzing the illustration in Fig. 2, which represents a connected component formed by intersection of four nuclei, and shows the respective points of relative minimum P_1 to P_6 . These points corresponds to the relative minima (or local minima) of the curvature $k(t)$ of the contour $C(t)$. The inset in Fig. 2 depicts the curvature $k(t)$ and the corresponding relative minima P_1 to P_6 .

The change of sign in the values of curvature $k(t)$ identifies the alternation between a point of relative maximum and a point of relative minimum of the curve. Between two points of relative maximum there is always a relative minimum, and vice versa. By using this analysis of maximum and minimum is possible to find the all the relative minima of the contour, and these points corresponds to the location of where the nuclei should be separated.

After the identification of the points of relative minimum, it is necessary to establish the correspondence between them in order trace the line which will separate the nuclei. In the proposed method, to select the pairs candidates to correspondence, the points of relative minimum are separated into two groups, accordingly to which side of the merged nuclei they belong. In order to define in which side of the merged block the points are located, the contour was divided into two regions. The two points of the contour which define these regions corresponds to the two peaks of the curvature of a oversmoothed version of the contour (i.e., the contour was smoothed by applying a Gaussian filter with $\sigma = 50$). This approach allows to identify the extremities of a merged block and it is translation and rotation invariant. For example, in Fig. 2, points P_1 , P_2 and P_3

are located on one side of the form, while P_4 , P_5 and P_6 are located on the other side.

The next step is to consider only the points located at one of the sides of the contour. For each of these points, the vectors \vec{V}_{p_i} are constructed considering the current point and all the points on the opposite side. Then, the angles θ_i formed between the normal vector \vec{V}_n at the current point with the respect to the vectors \vec{V}_{p_i} are computed. The smallest θ_i indicates to which point of the opposite side the line that splits the nuclei should be drawn. Consider, for example, the point P_1 in Fig. 2. Then, the angles θ_i formed between the normal vector \vec{V}_n at the point P_1 with the respect to the vectors \vec{V}_{p_1} , \vec{V}_{p_2} and \vec{V}_{p_3} are computed. The vectors \vec{V}_{p_1} , \vec{V}_{p_2} and \vec{V}_{p_3} are formed considering P_1 and its opposite points P_6 , P_5 and P_4 , respectively. For the considered example, the smallest θ_i corresponds to the angle formed between \vec{V}_n and \vec{V}_{p_1} , hence P_6 is identified as the corresponding point of P_1 and a line between these points is drawn in order to split the nuclei.

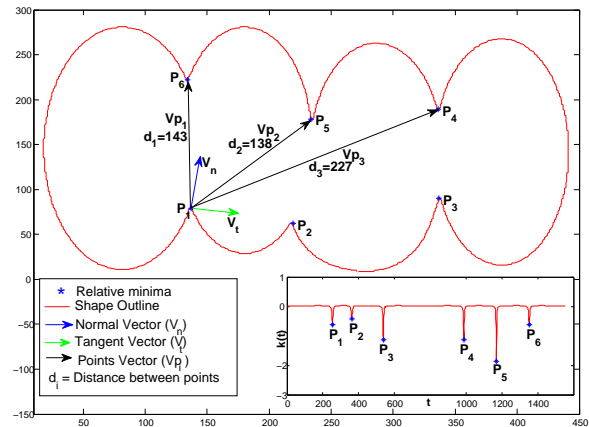


Figure 2: Illustration of a merged nuclei and the normal and tangent vector of one relative minimum point of the contour of the object. Points P_1 to P_6 are the relative minima of the contour. The vectors \vec{V}_{p_1} , \vec{V}_{p_2} and \vec{V}_{p_3} are formed considering the point of relative minimum P_1 and its opposite points P_6 , P_5 and P_4 , respectively. The distances d_i between P_1 and the opposite points are also shown. Note that the minimum distance cannot be used as a criteria to find the corresponding opposite point, as $d_2 < d_1$. In the inset it is depicted the curvature $k(t)$ of the shape of the object, along with the respective relative minimum peaks.

It is worth to note that the minimum distance between the opposite points could also be thought as a way of finding correspondence between opposite points. However, we found some examples where the smallest distance does not represent the desired correspondence. For example, in Fig. 2 the distance between the point P_1 and P_5 is lower than the distance between P_1 and P_6 . On the other hand, analysis of θ is a safe choice, since

the criterion of lowest θ precisely defines the opposite point corresponding to the current point.

Finally, after the correspondence between the relative minima points are established, a straight line between then is drawn in the image in order to separate the nuclei, i.e., the pixels under the line segment unifying opposite relative minima are set to zero, separating the nuclei. In this stage, it is possible to obtain the fully separated nuclei. This procedure is applied for all the points located on the same side of the first analyzed point (i.e., the points P_2 and P_3 in the example of Fig. 2). However, if after this first iteration two or more nuclei remains merged, the procedure can be reapplied in order to obtain the expected result.

4 EXPERIMENTS

In order to evaluate the proposed methodology, 10 sagittal images from different embryos were collected. All embryos are at the cleaved cycle 14A [Gilb03]. These images were processed and the obtained results were compared with other proposed techniques [Ferr97, Malp97, Kosm99, Bala12].

The parameters used in the pre-processing and image binarization steps are described as follows: the optimal size of the Gaussian filter usually depends on the size of the objects in the image. We found that a $\sigma=3$ and window size of 5×5 was adequate for subsequent analysis. The linear mapping function has been applied considering an angular coefficient equal to 9 and a linear coefficient -35 in the cases which were necessary to adjust the brightness and contrast of the image. For the Top-hat and Bot-hat transformations it was used a disk structuring element with radius 15 and 5, respectively. For the Gaussian smoothing used in curvature estimation (in the frequency space), a standard deviation equal to 5 was used. The following sections details the analysis performed.

4.1 Dataset

The images used in this work were obtained in a Leica TCS SP5 confocal microscopy, at the *Inmetro - National Institute of Metrology, Quality and Technology*. To visualize the *Drosophila* nuclei, we used the blue-fluorescent DAPI nucleic acid stain from Invitrogen. Each image has 1024×1024 pixels and 8 bits depth. Fig. 1(a-d) shows four of the analyzed images.

4.2 Results

In order to evaluate the accuracy of the proposed method, all 10 collected images were tested. These images contains about 61 to 183 nuclei, being 1452 in total.

Among the images collected, Fig. 1(i-l) shows the nuclear masks obtained for four images. In all images analyzed most of the nuclei were properly detected, producing a nuclear mask with a good quality.

A quantitative analysis was performed based in the number of detected nuclei. This was initially accomplished by counting this number based on visual inspection of original images. Posteriorly, the number of nuclei identified from the nuclear mask was compared to the original images. The results are summarized in Table 1, where it can be noted that the proposed method has, with reference to the number of nuclei, 94,4% of successfully detected nuclei.

Rightly detected nuclei	94,4%
Merged nuclei	3,9%
Absent nuclei	0,9%
Over-segmentation	0,8%

Table 1: Results of the application of the proposed method.

Most of the nuclei were properly identified, since only 3,9% of nuclei have not been properly separated (Merged nuclei). These were the case for blocks of merged nuclei without regions of well-defined valleys. This characteristic prevents the localization of the points of minimum curvature, therefore the segments cannot be separated.

A small portion of cell nuclei, about 0,9%, weren't accurately detected (Absent nuclei). This was the case for image regions with a very low contrast, where the Otsu's algorithm were unable to precisely identify the nuclei.

The rate of over-segmentation (i.e. individual nuclei that were split in more than one component) is very low (0,8%) and shows the efficiency of method in segmenting complex nucleus configurations. It is important to note that most of the errors verified were generated from images with low contrast and/or distinctness. Such images show nuclei with contours poorly defined (blurry), in addition to little variation of light intensities between regions of background and foreground (nuclei). These images are even difficult to perform manual segmentation of nuclei.

In addition, in order to assess the quality of the proposed segmentation method, we compared our results with the results obtained by techniques proposed by Costa et al. [Ferr97], Malpica et al. [Malp97], Kosman [Kosm99] and Bala [Bala12], which are all variations of watershed segmentation algorithm. The results of these methods are shown in Fig. 3 and are summarized in Tab. 2.

The choice to compare this work with the proposal of Kosman D. [Kosm99] was motivated by its application in the analysis of gene expression patterns of *Drosophila*. The method was used in the creation of FlyEx database [Pisa09], and works with superficial images of *Drosophila* embryos.

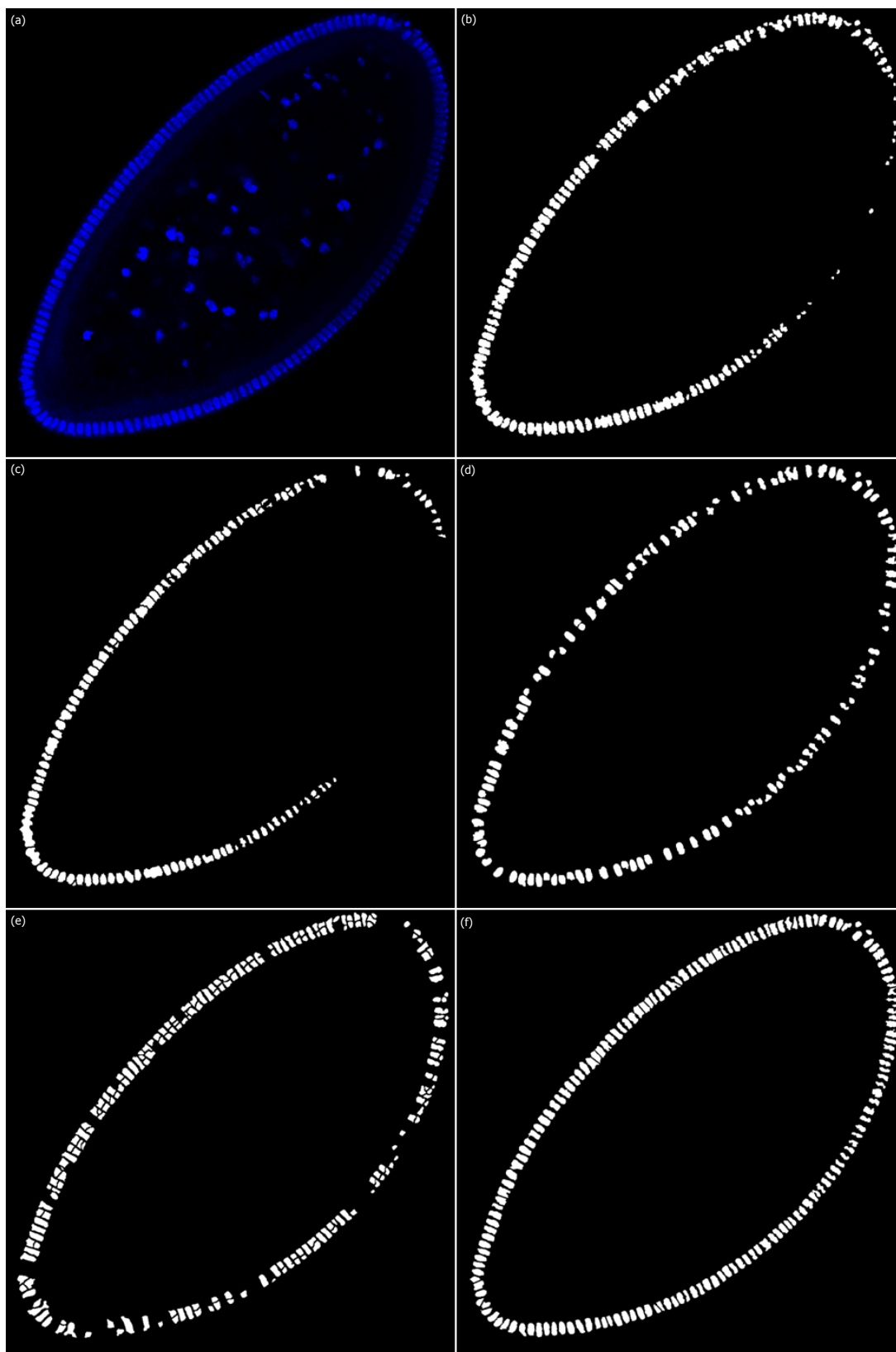


Figure 3: Comparative results of the segmentation obtained by different techniques. (a) Original image. (b) Segmentation result obtained by Costa et al. [Ferr97]. (c) Segmentation result obtained by Malpica et al. [Malp97]. (d) Segmentation result obtained by Bala [Bala12]. (e) Segmentation result obtained by Kosman D. [Kosm99]. (f) Segmentation result obtained by the proposed method.

	1-Our method	2- [Kosm99]	3- [Ferr97]	4- [Malp97]	5- [Bala12]
Rightly detected nuclei	1371 (94,4%)	937 (64,5%)	680 (46,8%)	982 (67,6%)	1043 (71,8%)
Merged nuclei	56 (3,9%)	48 (3,3%)	613 (42,2%)	192 (13,2%)	122(8,4%)
Absent nuclei	13 (0,9%)	200 (13,8%)	138 (9,5%)	247 (17,0%)	263 (18,1%)
Over-segmentation	12 (0,8%)	267 (18,4%)	21 (1,4%)	31 (2,1%)	24 (1,7%)

Table 2: Comparison of nuclei segmentation for different methods.

The comparison of the method proposed here with the works of Costa et. al [Ferr97], Malpica et al. [Malp97] and Bala [Bala12] are justified because they address revisions of Watershed algorithm, a method widely used to segment globular objects (such as nuclei/cells) in fluorescence images (2D or 3D) [Peng08, Chen12, Chan12, Du10, Hukk10, Clop10].

The Costa's method [Ferr97] (Fig. 3(b)), compared to methods 2, 4 and 5 (Table 2) showed the lowest rate of over-segmentation. However, it generated the largest number of merged nuclei, mainly in regions with weak borders between the nuclei.

The Bala's method [Bala12] (Fig. 3(d)), after our method, obtained good results in the metric 1 (Rightly detected nuclei (Table 2)). On the other hand, similarly to the Malpica's method [Malp97] (Fig. 3(c)), it showed large portion of nuclei absent (18,1% - Table 2). These binarization errors were largely due to variations in the nuclear signal intensity, specifically, the weak signal made it difficult the definition of markers for the Watershed algorithm.

The kind of errors mentioned above are, in some extent, caused by the choice of parameter settings. Thus, a clear difficulty with such algorithms is the effort required to tune them by selecting appropriate parameter settings to different images.

In general, the experimental results show that the proposed method is more effective and highly competitive, both qualitatively and quantitatively. Furthermore, we report differences in the quality of performance of the algorithms 2–5 (Table 2).

5 DISCUSSION AND CONCLUSION

In this paper it is proposed a new segmentation method for nuclei identification from sagittal images of *Drosophila* embryos. The differential of the method is the analysis of the curvature of binary shapes in order to segment complex nuclei configuration, such as merged nuclei.

Many approaches have been used to solve this problem in the last years, especially employing segmentation methods based on variations of watershed algorithm. However, some problems are recurrent when these algorithms are used in different applications, such as the difficulty to define the markers, the efforts required to define parameters and the over-segmentation.

The results with the proposed segmentation method demonstrated that it can segment nuclei from images of *Drosophila* embryos with higher efficiency when compared to other methods. The technique proposed also can be applied to segment similar configurations of cell nuclei in other kinds of images, such as nuclei/cells commonly found in fluorescence images, where no satisfactory solution was found yet.

The proposed method is efficient and accurate, and can be integrated in a database of sagittal images of *Drosophila* embryos, contributing to gain new insights from this biological model.

6 ACKNOWLEDGMENTS

B.A.N.T. thanks PROPP-UFU, CNPq and FAPEMIG (PEE-00436-13 and *Rede 52/11*) for financial support. D.J.S. thanks CNPq (process 153870/2011-7) for the financial support. F.J.P.L. thanks FAPERJ. P.M.B. would like to thank FAPERJ and CNPq. M.A.C thanks CAPES and CNPq.

7 REFERENCES

- [Attn54] F. Attneave. "Some Informational Aspects Of Visual Perception". *Psychological Review*, Vol. 61 (3), pp. 183–193, 1954.
- [Bala12] A. Bala. "An Improved Watershed Image Segmentation Technique Using MATLAB". *International Journal Of Scientific and Engineering Research*, Vol. 3, No. 6, pp. 1206–1206, 2012.
- [Bele05] M. E. Beletti, L. F. Costa, and M. P. Viana. "A spectral framework for sperm shape characterization". *Computers in Biology and Medicine*, Vol. 35, No. 6, pp. 463–473, 2005.
- [Beuc79] S. Beucher and C. Lantuejoul. "Use of watersheds in contour detection.". In: *Internat. Workshop on Image Processing, Real-time Edge and Motion Detection/Estimation*, Vol. 132, pp. 1–12, 1979.
- [Chan12] Y.-K. Chan, P.-Y. Pai, C.-C. Liu, Y.-S. Wang, C.-W. Li, and L.-Y. Wang. "Fluorescence Microscopic Image Cell Segmentation". *International Journal of Future Computer and Communication*, Vol. 1, pp. 72–75, 2012.

- [Chen12] S. Chen, M. Zhao, G. Wu, C. Yao, and J. Zhang. “Recent Advances in Morphological Cell Image Analysis”. *Computational and Mathematical Methods in Medicine*, Vol. 2012, p. 10, 2012.
- [Clop10] F. Cloppet and A. Boucher. “Segmentation of complex nucleus configurations in biological images”. *Pattern Recogn. Lett.*, Vol. 31, No. 8, pp. 755–761, 2010.
- [Cost09] L. F. Costa and R. Cesar Jr. *Shape classification and analysis: theory and practice*. CRC Press, 2 Ed., 2009.
- [Crow12] D. C. Crowther and K.-F. Chen. “Functional genomics in Drosophila models of human disease”. *Briefings in Functional Genomics*, Vol. 11(5), pp. 405–4115, 2012.
- [Du10] X. Du and S. Dua. “Segmentation of fluorescence microscopy cell images using unsupervised mining”. *The open medical informatics journal*, Vol. 4, pp. 41–49, 2010.
- [Ferr97] J. A. Ferreira Costa, N. D. A. Mascarenhas, and M. L. de Andrade Netto. “Cell nuclei segmentation in noisy images using morphological watersheds”. *Proc. SPIE, Applications of Digital Image Processing XX*, Vol. 3164, pp. 314–324, 1997.
- [Fowl08] C. Fowlkes, C. L. Luengo Hendriks, S. Keranen, G. Weber, O. Rubel, M.-Y. Huang, S. Chatoor, A. DePace, L. Simirenko, C. Henriquez, A. Beaton, R. Weizmann, S. Celniker, B. Hamann, D. Knowles, M. Biggin, M. Eisen, and J. Malik. “A Quantitative Spatiotemporal Atlas of Gene Expression in the Drosophila Blastoderm”. *Cell*, Vol. 133, No. 2, pp. 364–374, 2008.
- [Gilb03] S. Gilbert. *Developmental Biology*. Sinauer Associates, 7 Ed., 2003.
- [Gonz08] R. Gonzalez and R. Woods. *Digital Image Processing*. USA, 3 Ed., 2008.
- [Greg07] T. Gregor, E. F. Wieschaus, A. P. McGregor, W. Bialek, and D. W. Tank. “Stability and Nuclear Dynamics of the Bicoid Morphogen Gradient”. *Cell*, Vol. 130, pp. 141–152, 2007.
- [Houc02] B. Houchmandzadeh, E. Wieschaus, and S. Leibler. “Establishment of developmental precision and proportions in the early Drosophila embryo”. *Nature*, Vol. 415, No. 6873, pp. 798–802, 2002.
- [Huan08] M.-Y. Huang, O. Rübél, G. Weber, C. Hendriks, M. Biggin, H. Hagen, and B. Hamann. “Segmenting Gene Expression Patterns of Early-stage Drosophila Embryos”. In: L. Linsen, H. Hagen, and B. Hamann, Eds., *Visualization in Medicine and Life Sciences*, pp. 313–327, Springer Berlin Heidelberg, 2008.
- [Hukk10] J. Hukkanen, A. Hategan, E. Sabo, and I. Tabus. “Segmentation of cell nuclei from histological images by ellipse fitting”. In: *Proc. of the European Signal Processing Conference, Aalborg, Denmark*, pp. 1219–1223, 2010.
- [Jaeg04] J. Jaeger, M. Blagov, D. Kosman, K. N. Kozlov, Manu, E. Myasnikova, S. Surkova, C. E. Vanario-Alonso, M. Samsonova, D. H. Sharp, and J. Reinitz. “Dynamical Analysis of Regulatory Interactions in the Gap Gene System of Drosophila melanogaster”. *Genetics*, Vol. 167, No. 4, pp. 1721–1737, 2004.
- [Jans05] H. Janssens, D. Kosman, C. Vanario-Alonso, J. Jaeger, M. Samsonova, and J. Reinitz. “A high-throughput method for quantifying gene expression data from early Drosophila embryos”. *Development genes and evolution*, Vol. 215, No. 7, pp. 374–381, 2005.
- [Kosm99] D. Kosman, J. Reinitz, and D. H. Sharp. “Automated assay of gene expression at cellular resolution”. In: *Proceedings of the 1998 Pacific Symposium on Biocomputing*, pp. 6–17, 1999.
- [Kozl08] K. Kozlov. “ProStack: a new platform for image analysis”. In: *CSHL Conference "Computational Cell Biology"*, 2008.
- [Levi85] M. D. Levine. *Vision in man and machine*. McGraw-Hill, 1985. NY.
- [Luen06] C. L. Luengo Hendriks, S. V. E. Keränen, C. C. Fowlkes, L. Simirenko, G. H. Weber, A. H. DePace, C. Henriquez, D. W. Kaszuba, B. Hamann, M. B. Eisen, J. Malik, D. Sudar, M. D. Biggin, and D. W. Knowles. “Three-dimensional morphology and gene expression in the Drosophila blastoderm at cellular resolution I: data acquisition pipeline.”. *Genome Biol*, Vol. 7, No. 12, p. R123, 2006.
- [Malp97] N. Malpica, C. O. de Solórzano, J. J. Vaquero, A. Santos, I. Vallcorba, J. M. García-Sagredo, and F. D. Pozo. “Applying watershed algorithms to the segmentation of clustered nuclei”. *Cytometry*, Vol. 28, No. 4, pp. 289–297, 1997.
- [Otsu79] N. Otsu. “A threshold selection method from gray-level histograms”. *IEEE Trans-*

- actions on Systems*, Vol. 9, pp. 62–66, 1979.
- [Pan06] J.-Y. Pan, A. G. R. Balan, E. P. Xing, A. J. M. Traina, and C. Faloutsos. “Automatic mining of fruit fly embryo images”. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 693–698, ACM, New York, NY, USA, 2006.
- [Peng08] H. Peng. “Bioimage informatics: a new area of engineering biology”. *Bioinformatics*, Vol. 24, No. 17, pp. 1827–1836, 2008.
- [Pisa03] A. Pisarev, E. Poustelnikova, M. Samsonova, and P. Baumann. “Mooshka: a system for the management of multidimensional gene expression data in situ”. *Information Systems*, Vol. 28(4), pp. 269–285, 2003.
- [Pisa09] A. Pisarev, E. Poustelnikova, M. Samsonova, and J. Reinitz. “FlyEx, the quantitative atlas on segmentation gene expression at cellular resolution”. *Nucleic Acids Research*, Vol. 37, No. suppl 1, pp. D560–D566, 2009.
- [Pous04] E. Poustelnikova, A. Pisarev, M. Blagov, M. Samsonova, and J. Reinitz. “A database for management of gene expression data in situ”. *Bioinformatics*, Vol. 20, No. 14, pp. 2212–2221, 2004.
- [Quel10] P. Quelhas, M. Marcuzzo, and A. M. Mendonça. “Cell Nuclei and Cytoplasm Joint Segmentation Using the Sliding Band Filter”. *IEEE Transaction on Medical Imaging*, Vol. 29, No. 8, pp. 1463–1473, 2010.
- [Rbel06] O. Rübél, G. H. Weber, S. V. E. Keränen, C. C. Fowlkes, C. L. L. Hendriks, L. Simirenko, N. Y. Shah, M. B. Eisen, M. D. Biggin, H. Hagen, D. Sudar, J. Malik, D. W. Knowles, and B. Hamann. “Pointcloudxplore: Visual analysis of 3d gene expression data using physical views and parallel coordinates”. In: *Eurographics/IEEE-VGTC Symposium on Visualization Proceedings*, pp. 203–210, 2006.
- [Shap01] L. G. Shapiro and G. C. Stockman. *Computer Vision*. Prentice Hall, New Jersey, 2001.
- [Shih09] F. Y. Shih. *Image Processing and Mathematical Morphology: Fundamentals and Applications*. CRC Press, 2009.
- [Soil99] P. Soille. *Morphological Image Analysis: principles and applications*. Springer-Verlag Berlin Heidelberg, 1999.
- [Solo10] C. Solomon and T. Breckon. *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab*. Wiley-Blackwell, 2010.
- [Surk08] S. Surkova, E. Myasnikova, H. Janssens, K. N. Kozlov, A. A. Samsonova, J. Reinitz, and M. Samsonova. “Pipeline for acquisition of quantitative data on segmentation gene expression from confocal images.”. *Fly (Austin)*, Vol. 2, No. 2, pp. 58–66, March 2008.
- [Toma07] P. Tomancak, B. P. Berman, A. Beaton, R. Weiszmann, E. Kwan, V. Hartenstein, S. E. Celniker, and G. M. Rubin. “Global analysis of patterns of gene expression during *Drosophila* embryogenesis.”. *Genome Biol*, Vol. 8, No. 7, p. R145, 2007.
- [Vinc91] L. Vincent and P. Soille. “Watersheds in digital spaces: An efficient algorithm based on immersion simulations”. *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 13, No. 6, pp. 583–598, 1991.
- [Xion06a] G. Xiong, X. Zhou, and L. Ji. “Automated segmentation of drosophila RNAi fluorescence cellular images using deformable models”. *IEEE Transactions on Circuits and Systems*, Vol. 53, No. 11, pp. 2415–2424, 2006.
- [Xion06b] G. Xiong, X. Zhou, L. Ji, P. Bradley, N. Perrimon, and S. Wong. “Segmentation of *Drosophila* RNAi Fluorescence Images Using Level Sets”. In: *Image Processing, 2006 IEEE International Conference on*, pp. 73–76, 2006.
- [Yan08] P. Yan, X. Zhou, M. Shah, and S. T. Wong. “Automatic Segmentation of High-Throughput RNAi Fluorescent Cellular Images”. *IEEE Transactions on Information Technology in Biomedicine*, Vol. 12, No. 1, pp. 109–117, 2008.
- [Zhan12] C. Zhang, C. Sun, R. Su, and T. D. Pham. “Segmentation of clustered nuclei based on curvature weighting”. In: *Proceedings of the 27th Conference on Image and Vision Computing New Zealand*, pp. 49–54, ACM, New York, NY, USA, 2012.

Computational Celtic Canvas for Zoomorphs and Knotworks

Richard B Doyle and SK Semwal

Department of Computer Science
University of Colorado at Colorado Springs
Colorado, 80918, USA
rbdoyle@pcsys.net ssemwal@uccs.edu

ABSTRACT

This paper presents *Celtic Canvas* -- a framework towards computationally generating patterns similar to simple Celtic artwork. As Celtic patterns are highly structured, a robust model of curvature dynamics is introduced in the paper to allow the evolution of space curves as a foundation for line drawing and structural anatomy. Cellular automata integrates several features including -- lines and forms, allocation of space, a solution of crossing states, the occlusion processing, depth cues, and line weighing. Lattice gas automata assist in generating varying width patterns. Original digitally available artwork provides shapes and forms as line-art. Our novel contribution is that we are successful in generating examples of knotwork and zoomorphs (animal designs shapes) that mimic the characteristics of the Celtic art forms. Future research areas are also identified.

1. INTRODUCTION

Reconstruction of Celtic design methods could perhaps be traced to the work of J. Romilly Allen's [All93] 1903's survey which was followed by the studies of George Bain [Bai51] in 1951. As part of her survey of Celtic interlace in Northumbria (Northern England, 500-1100), Gwenda Adcock [Adc74] developed the system now used for archeological description and analysis. Iain Bain in 1986 [Bai86] transformed many of his father's construction techniques into concise procedures or algorithms. Peter Cromwell [Cro93] followed with his examination of frieze patterns, and is credited with considering interlace as being traced by a ray reflected by bounding structure [Fun07]. Paul Gerdes described this generator in detail, first for Tchokwe and Tamil pictograms [Ger90] and later for Celtic interlace [Ger99]. A theory of mirror curves emerged in [Jab95]. Adcock's research [Adc74] discussed evidence of templates. While the approach is workable in typography, it also helped illustrate another of Adcock's observations, that a skilled artist is needed to prevent monotonous patterns. Christians Mercat [Mer97] observed that crossings in Celtic interlace comprise an encoding (tetravalent, of four converging cord segments) enabling the specification of arbitrary interlace in terms of planar graphs. Frank Drewes [Dre89] demonstrated graph grammars where terminal symbols are associated with tiles. This resembles the approach taken in Lindenmayer-Systems [Pru90] as well. One of the characteristics of the illustrations is to appear hand drawn. So the following guidance was developed: (a) reduce artifacts that may be perceived as the result of a

rendering process, (b) portray materials accurately according to context; (c) understand what to emphasize; (d) avoid patterns and regularities; (e) follow established rules of traditional rendering techniques – how the lines are placed and use visual cues such as line weight and depth variation.

Development in the theory of alternating knots [Bro05, Chu05] and tangles now extends to classification and enumeration [Bae07]. Adcock [Adc02] completed a study of interlaced animal designs (zoomorphs) in Bernician sculpture and their relationship to work produced by the monastery at Lindisfarne. One significant difference between zoomorphs and interlace is that interlace involves strictly closed curves and animal designs more often involve open curves. The Isenberg [Ise06] study recommends experimentation with approaches to line weight and cuing and line shape as that can convey emotion [Fre03]. Knotwork, such as work by David Llewellyn-Jones, posted on the web, does not have a zoomorph implementation. NPR research today includes modeling for pencil [Mer08] and ink on paper [Chu05], brush and paint on canvas [Bax04], medieval manuscripts [Bro04] and collections at Trinity College Dublin [Qui07]. Parametric curves, such as Bezier, Hermite, or B-Splines [Far97] tend to be less expressive than those drawn by hand, yet managing continuity is always an important piece which may affect the artwork. Work by Kurt Fleischer [Fle97] on self-organizing geometry is relevant for generating adaptive and curved lattice gas models [Har71], and can be useful due to the evolution of a trefoil shaped filament [Mal96] shapes and shape grammars. In this paper, we focus on providing research on how to computationally generate knotwork [LAr07] and zoomorphs (animal design forms). Our research and results are novel and promising in the sense that we have been successful in capturing some of the innate aesthetics and simplicity observed in knotwork and zoomorphs of Celtic Art form. These are shown as crossing (Figures 2, 4, 12, and 13), design elements (Figure 6-8), and zoomorphs (Figures 9 and 14). These are basic elements found in Celtic Art, and according to best of our

□ Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

knowledge, not captured by any other existing computational method.

2. BACKGROUND

To conform to the style of insular design from the 9th century, proportions for interlace are found in Adcock [Adc74; 02]. Alternatively these can also be measured from lithographs, photography from the manuscript departments of universities, or measured from digital photography available on the web. Similar measurements may be developed for zoomorphs, plant designs, humans, key patterns, spirals, and so forth.

The key idea in complex systems theory is that local interactions give rise to organized global behavior. Brush-strokes by an artist are examples of local control creating global emergence, which is the overall painting. Modeling of these systems involves simulation over assemblies of discrete elements such as brush-strokes. Broad surveys have been done [Fle95]. Organizations include lattice gas automata (LGA) [Har76]. Other related work are in [Sim94] but for very different applications.

Differential Geometry

The nib of a pen traces a line on a writing surface. The point of contact may be thought of as a particle moving along a space curve in the plane of the writing surface. From the fundamental theorem of space curves, if curvature and torsion are differentiable over the arc length and curvature is everywhere positive, there exists a unit speed space curve with those properties. The initial unit tangent and unit normal vectors may be assigned as long as their dot product is everywhere zero. For a planar space curve, torsion is everywhere zero. In this special case, the curvature function may be signed, allowing turns left and right. The formulation is interesting to us as varying shape curve can be generated given an initial position X, unit tangent (T), unit normal (N), and B, and assigned curvature K(s) and torsion (TO(s)) values which provide the rate of change of present position of the curve using curve parameter (s) based on differential equation formulation described in [Gra98].

Mirror Curve Generator

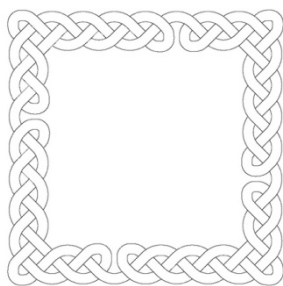


Figure 1: Single knot three cord border.

Part of the fascination of mirror curves in design is that an outcome may not be immediately obvious. For example, while placing breaks in a variation on a grid studied by

Aidan Meehan [Mee03], it was not apparent that the preceding knot-work in Figure 1 uses a *single* cord. Designs in this study, following the example of George Bain, are first developed using medial lines with a density of a one cord per square. As a cord passes through a square, its departure is designated by a chain code. Chain code directions on the Moore neighborhood are shown in Figure 2.

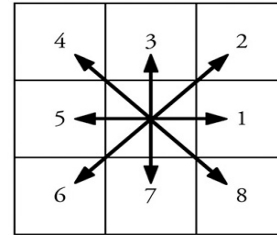


Figure 2: chain code directions.

Using a varying grid, an automaton, called a lattice gas automaton, for left reflection is derived by examining part of the following knot in the following work (Figure 3) using chain codes:

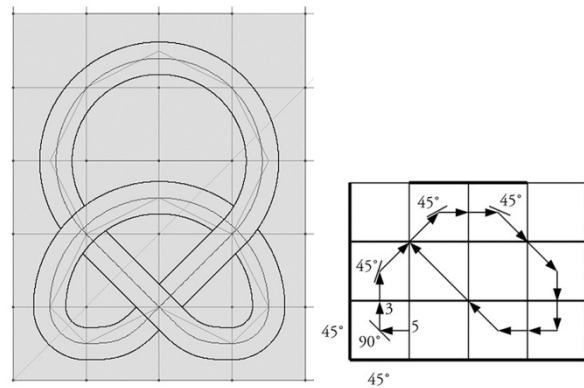


Figure 3: (left) Derivation of left mirror (bottom 3 by 4) moves as (Right) chain codes (3 by 4).

For left reflection assume that mirrors are placed to the left of the particle path in the lattice gas particle automaton. Begin with the particle in the lower left corner heading in chain code direction five (Figure 3). The model describes a particle that interprets a break as a 45-degree direction change. Angles are additive. Addition of random moves can provide more variety as the particle moves, offering variation of patterns.

The motion of the particle simulates the generation of a mirror curve as chain codes. Each mirror curve defines the intersections of a guide curve with the grid. If the path traced by the particle is closed in the desired curve, Thurston's theorem says that an alternating knot covering that path exists. A GCA particle tracing the edges of the major grid, interpreting breaks as mirrors, can use Mercat's algorithm to find a coding for the crossings (Figure 4). Alternatively, the encoding can be determined using a mobile cellular automaton that traces an image of the

solution for each guide curve across the drawing surface to interpret a composite image of all the guide curves in the composition.

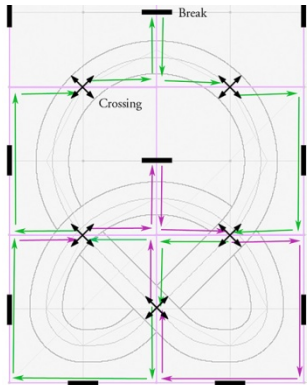


Figure 4: A path traced by Mercat's coding algorithm.

From this all of the intersections are identified. Each intersection, as with Mercat's algorithm, is used to hold the identification of guide curves that pass through it and the reservation of the above or below state for each intersection of the knot. If, while traversing the image of a guide curve, the mobile automaton encounters an intersection where the needed reservation is already taken, the solution fails. In the case of designs involving only knots, resolution of these failures may be resolved through a divide and conquer strategy. However, if the design contains tangles, a solution may not exist. Where a crossing solution exists, it is encoded as a function of arc length on each guide curve in the composition. The coding is adjusted so that state transitions occur half way between intersections.

The lattice gas particle path is drawn as a polygon joining the centers of cell entry and exit points. This automatically generates results in Glassner's visualization [Gla99]. Since the grid is square, the generating sequence for a space curve covering the path can be assembled by collecting sequences from a small set of standard curve segments, indexed by which turn is taken by a particle on exiting a cell relative to the direction on entry. When the grid is irregular, a suitable curve for each segment must be adaptively produced. For example, models for the curve segment are selected and then, in the worst case, fitted by evolutionary search on the ranges of their parameters and the one with the best-fit wins. Our observation is that the behaviour of the particle with such set interfaces resembles what an artist may be doing when judging how to draw a line. In studying the lines of Celtic interlace some of these shapes are familiar. Their incorporation may also be important as suggested by a study on the effect of line shapes on the perception of emotion or style [Fre03].

3. INFLATING THE CURVES

Once the lines representing the Celtic-canvas are satisfactory, closed or open curves represented by cords or the lines can be inflated. For interlace, a cord must be fitted to the guide curve. This process is often described as inflating the cord. If the left and right edges of the cord are

assumed to be lines parallel to the guide curve, the parallel line theorem from differential geometry may be used [Gra98]:

$$c_{\text{inflated}}(\alpha, s, t) = \frac{sJ\alpha'(t)}{\|\alpha'(t)\|}$$

Tangent velocity of the curve is normalized at t , and scaled by distance s ; J is the two-dimensional Jacobian [Gra98; Doy08]. The cord is inflated in the direction of the curve's normal and then repeated in the opposite direction to create cords from lines in a cylindrical fashion. The radius of the cord is used as the scale factor and can vary based on parameter t as shown above. Parameter t could also be used for mapping textures in future.

Design Elements as Forms and Sleeves

Design elements are described as forms and sleeves, as detailed in [Doy08]. Each is represented by line art for the shape of a given element around symmetries to be provided by guide curves which are generated from the given celtic art piece as explained below. Forms are used primarily for zoomorphs and plants in the Celtic Art. Sleeves are specialized forms that provide the shape for cords in the knotwork. The reason for using line art is to allow a designer the freedom to draw these shapes as needed without relying on any particular technology. These line art can then be inflated to create knotwork supplementing the computer generated forms. Design elements provide an efficient mean to provide templates, and can be used as tiles to create continuous curves if properly used.

A form (input shape) is specified in terms of four files: anatomy, symmetry, segmentation, and pivots. Figure 5 shows such an arrangement for our implementation. Anatomy defines what a form looks like (leftmost Figure 5). Symmetry defines a skeleton (left-middle Figure 5). Segmentation defines where a form can be split and stretched along symmetry (right-middle Figure 5). Pivots define where parts of the anatomy may be independently turned and are left for future study (rightmost Figure 5).

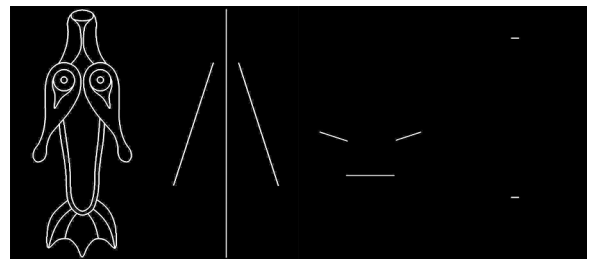


Figure 5: Celtic Snake anatomy, symmetries, segmentation and pivots (left to right).

Extraction of lines from the anatomy drawing is supported by connected region enumeration and a simple set algebra, so desired lines can be obtained (Figure 6). Extracted lines are sampled as distance functions along the arc length of the corresponding symmetry and then coded as part of the genotype for the shape. To resolve occlusion during rendering, lines that form the outermost outline of the form

are marked as exterior. Sleeves are handled similarly and only have one symmetry. Sleeves are fitted to cover a portion of the arc length of a guide curve. Examples of cord sleeves with their specifications are as follows (Figure 7).

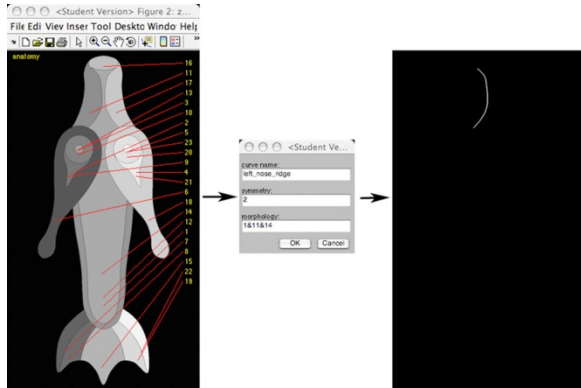


Figure 6: Left Nose ridges extracted from anatomy.

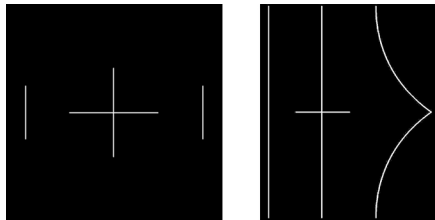


Figure 7: Chord sleeve, left, chevron sleeve right turn, right.

The well structured basic shapes representing genotypes (code for the line-art) for a composite sleeve are assembled on request when binding to the guide curve for a knot. Generative system implementations where genotype or binary bit pattern representing the eventual shape can be expanded to present line art (phenotypes) has been also used in our implementation. An appropriate sleeve (cord, left or right chevron) is selected for each segment of arc length of the guide curve, where a segment is the portion of the guide curve that passes through each cell in the grid.

Artifacts of hand drawn line and rendering

Noise or small random variations in basic line-art code shape or genotype, are used in our implementation to create variety of shapes or phenotypes which then can be represented in line form and inflated as explained earlier. Some of the issues identified by Isenberg [Ise06] have been included in this study. Vector noise accounts for variations in the movement of a designer’s hand when drawing. We assume that variation in the motion of the hand, perpendicular to the direction in which a line is being drawn, varies as an arrival problem. In other words random variation occurs at some rate, and can be modeled as an exponential distribution. The scaling term from the Brownian Bridge is borrowed to assure that the ends of walks meet as follows:

$$W_s = \sqrt{\frac{s(s_1 - s)}{s_1}} \cdot f(N_{0,1}) \quad 0 < s < s_1$$

where s_1 is the length of the respective walk and s is the parameter within the walk. This is implemented by genetically generating piecewise continuous curves whose positions left and right of zero are randomly varied by the normal distribution at the rate of the exponential distribution. The generator joining two points is zero seeking. These functions are expensive (time consuming) to generate; so, they are kept short and looped as needed. This noise process is added to the widths of lines centered on guide curves:

$$w_{noisy}(s) = w(s) + W_s$$

The lengths of these noise functions, the amount of their variation, and the rate at which variations occur are set by the designer. The above value of w provides variation of cord width simulating human hand drawing. The depth cue parameter is implemented as a multiplier, $d(s)$, on the width function. Again the curvature dynamics system is used to develop easements in and out of these cues. The combined effect of depth cue and hand motion variation becomes:

$$w_{noisy \& \text{depth}}(s) = d(s) \cdot w(s) + W_s$$

The designer can set an arc length distance from the guide curve being crossed to start a depth cue, as well, as separate parameters for the width of above and below states. The two pass rendering algorithm is implemented. The first pass draws all lines having the *above* state. Lines with the *below* state are drawn on the second pass. When the image of an exterior line having the above state is encountered on the second pass, the state of the lower line is changed to “occluded.” This state continues until the next exterior line of the form being passed under is reached. The Sousa pencil model [Sou03] simulates the abrasion and deposition of a graphite source when rubbed against paper. We also implemented an alternate approach for deriving guide curves as patterns generated from a genetic interpretation of a Perlin [42] noise process (Figure 8).

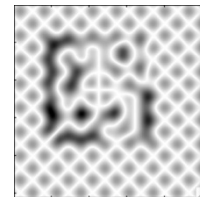


Figure 8: Cross in circle pattern using Perlin Noise.

A number of exotic problems arise first in attempting to find these patterns and then extract them for use in assembling interlace. Although promising, more is needed to computationally extract zoomorphs and knot-work from Perlin noise, especially representing the aesthetics and complexity of Celtic art. More details of this work are in [Doy08].

4. IMPLEMENTATION

Based on methods from the previous Section, our algorithm is summarized as follows:

- 1: Define structures (grids) with breaks on a canvas
- 2: Define design elements for the grids
- 3: Arrange and pose elements within the structure
- 4: Solve alternating crossings for knots and tangles, ensuring continuity as the curve crosses grid elements
- 5: Render a result

Multiple grids provide structures for local interactions that are consistent with neighbors joining two points across grid elements, thus providing consistency across grid boundaries. Quadrilateral meshes for structure grids were based on the templates of Matthias Muller-Hannemann [Mul97]. Design elements provide the context and include chain-code shapes (genotypes) or line-art from Celtic art pieces as guidance curves. Complete Celtic-Canvas system diagram is shown in Figure 9.

The system is organized around a set of user interface tools. Inputs to the system include mattes of the drawing surfaces(s), under drawings (optional), line drawings of forms (animals, plants, etc.), and of sleeves. Morphological analysis is used to provide an initial interpretation of drawing surfaces. Tools are provided for defining and editing grids and placing breaks, assisting with the design of guide curves (lattice gases and freehand drawing) and applying them to posing forms.

Celtic Canvas provides an opportunity to view such an art form as a complex system using the work of Mitchell

[Mit09]. Complex forms are simulated using mutation and variation of genotypes, and rendering them as phenotypes. Specifically, The forms and guide curves are easily identified as individuals (genotypes). Their phenotypes consist of both structural and visual components including variation of depth and width. Design elements are first formed by a skeleton of guide curves which are derived from their symmetries. The chromosome of a composite sleeve might contain hundreds of sequences, including the description of each thing to be drawn and it's binding to a guide curve. The genotype of a form provides everything needed to solve and pose its anatomy on the supporting guide curve(s).

A Celtic-composition in this system presented contains grids and design elements. It contains a constructor (ribosome) that builds phenotypes of its elements under the guidance of the designer. Its phenotype is the rendering of its design-elements which are results of our implementations as presented in Figures 1, 5, 8, and 10-14. The reason that the shapes generated by our implementation resemble Celtic-art forms is because design elements used in our system are either guided by Celtic-Art form or are designed that way.

Although our focus has been to computationally generate Celtic knotwork and zoomorphs, the formulation presented in Section 2 is also capable of producing three-dimensional shapes and forms if we use 3D design elements and suitable phenotypes. In addition, the formulation presented in this paper can be extended to *non-existent* new interpretations of Celtic-Art forms based on 3D Space filling arbitrary shapes using either solid or hollow three-dimensional cords and zoomorph as design elements.

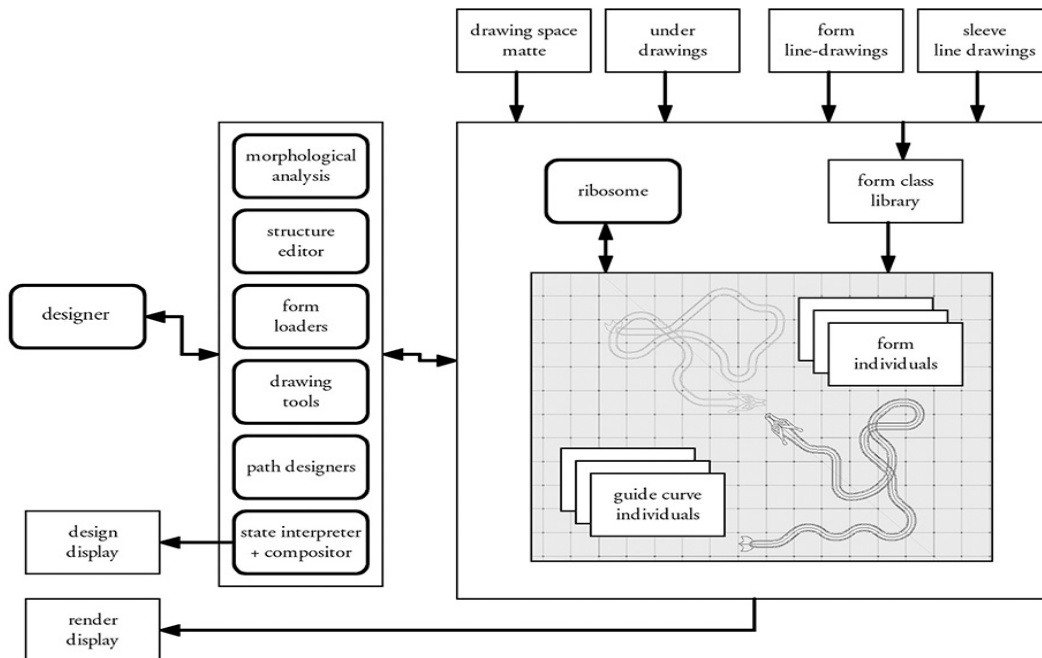


Figure 9: Celtic-Canvas: system diagram.

5. RESULTS

This section describes results of our implementation using Celtic-Canvas (Figure 10 [Doy08]). They are presented with examples from the Book of Kells and the Lindisfarne Gospels [Hen74, Mee94]. The primary references for comparison with hand-drawn examples are those developed by Adcock and the illustrations of Iain Bain. Our review of the literature found that automatic means for evaluating the quality of a visual design remains an open question [Ise06].

Our results in Figures 10-14 show major improvement over existing computational methods presented in [Gla99, Kap03], and take few minutes as symmetry, shape, skeleton and pivot points are defined, as discussed in [Doy08]. Since curvature is such a natural attribute of space curves, one of our first attempt was a Celtic spiral. Our reconstruction (left) uses a progression based on the golden ratio indicated (right) in Figure 11. The design was inspired by a triple spiral on folio 34r of the Book of Kells. It is presented here (left) without artifacts and in the original (Figure 10).



Figure 10: Triple spiral from MS 58 f34r, Board of Trinity College Dublin (right) with permission.

Lindisframe Knotwork (Bain): Although knotwork is not part of the lozenge in folio 34r of the Book of Kells, it is fundamental to most Celtic design. Using Celtic Canvas, a reconstruction using Celtic Canvas of a design from the Lindisfarne Gospels (f. 11) [7] is shown in Figure 11 using with (right) and without (left) slight artifact. Both are produced with our implementation.

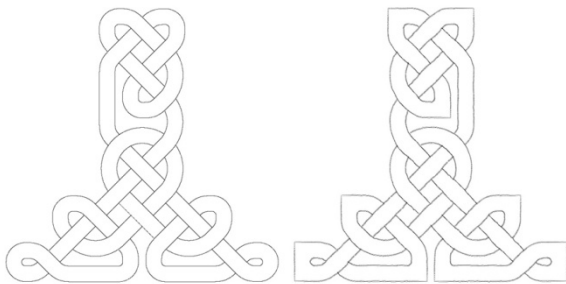


Figure 11: Lindisfarne (f. 11) reconstruction, left, and with added artifacts, right.

Lindisframe Knotwork (Adcock)

A more complex example of knotwork is a fragment from the Lindisfarne Gospels (f. 26) [Adc73, Bai51].

The design was not drawn from a single cord, but rather three. An omitted break revealed a design using only two cords. A single cord would be Celtic symbolism for eternity. A reconstruction with chevrons and artifacts is in Figure 12. Patterns for curves with larger radii can be recognized when curve segment sequences are symbolically coded. This figure includes curve sweetening by substitution of larger radii curves. These beautiful computationally generating curves using Celtic-Canvas can be further improved by evolving the parameters that affect their proportional fit in the grid.

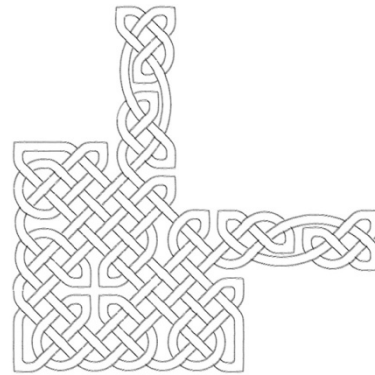


Figure 12: Lindisfarne Gospels (f. 26) corner with artifacts

Reptile (snake) based from Lozenge of MS 58 f 34r: A motivator for our research was the lozenge in cross of the Chi in folio 34r of the Book of Kells, first studied by George Bain in 1951[Bai51]. An example of a snake posed on a guide curve designed by a mirror curve using Celtic Canvas is in Figure 13.

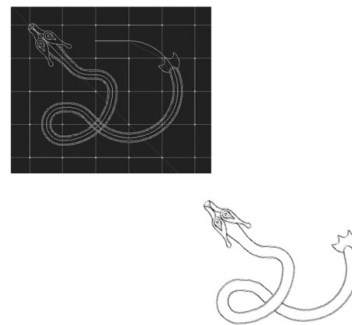


Figure 13: Posed reptile with artifacts

A significant portion of the design in the lozenge in folio 34r is devoted to a tangle of creatures. A series of reptiles were assembled as a test of the solver for crossings in alternating knots and tangles. An LGA (Lattice Gas Automata) with a higher degree of freedom was used to design guide curves to pose each of the snakes. The result of our implementation without artifacts is in Figure 14 and shows complex tangled designs Celtic Canvas can generate, specially

the visual variation in each resulting snake is interesting.

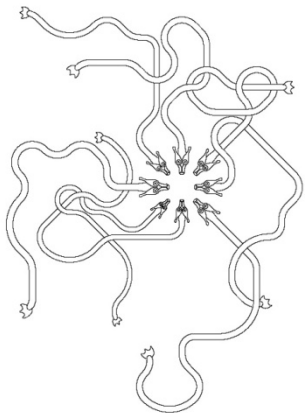


Figure 14: Tangle of reptiles.

6. CONCLUSIONS

In this paper, we have presented a novel implementation to computationally generate interlacing knotwork and zoomorphs seen in historical Celtic Art. The complex system based implementation integrates some of the aesthetics and characteristics of the original art form as its design elements. Significant contribution of our work is that shapes and forms, mimicking the original art, emerge as shown by several examples of knotwork (Figures 11, 12) and zoomorphs (Figure 14). In addition, we also developed a solution for alternating crossings and rendering. An implementation of additive vector noise based on our research provides hand-drawn quality to our work. We were not able to computationally generate complexities in Figure 15. George Bain's reconstruction of the lozenge at the crossing of strokes in the Chi of folio 34r in the Book of Kells, the XPI monogram page (Figure 15), is indeed remarkable and worth pursuing further. Appearing as a mass of tangles, it contains little of the regular structure found in Celtic interlace and would require further study.

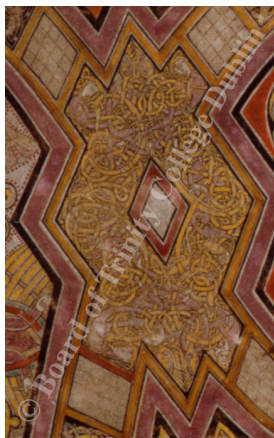


Figure 15: Lozenge of MS 58 f34r, Board of Trinity College Dublin with permission.
Full papers proceedings

Computationally generating and measuring the aesthetic quality of generated art form [Yev02] and identifying it to be similar to Celtic-art, and occlusion shading is another research challenge for the future. A line of inquiry, which is of interest to us, is the work by Itti, Dhavale and Pighin [Itt04] and Santella [San05]. These will be further investigated and are still open areas of research at this time.

7. REFERENCES

- [Abb98] Abbot, S. 1998. Steve Abbot's computer drawn Celtic knotwork, Knots3D, retrieved on October 29, 2005, from <http://www.abbott.demon.co.uk/knots.html>.
- [Adc74] Adcock, G. 1974. A Study of the Types of Interlace on Northumbrian Sculpture, MPh thesis, Durham University.
- [Adc02] Adcock, G. 2002. Interlaced Animal Design in Bernician Stone Sculpture Examined in the Light of Design Concepts in the Lindisfarne Gospels, PhD thesis, Durham University.
- [All03] Allen, J. and Anderson, J. 1903. The Early Christian Monuments of Scotland, the Society of Antiquaries of Scotland, republication by The Pinkfoot Press, 1993.
- [Bae07] Bae, Y. and Morton, H. 2007. The spread and extreme terms of Jones polynomials, Journal of Knot Theory and its Ramifications, 12, 359-374.
- [Bai51] Bain, G. 1951. Celtic Art: The Methods of Construction, William Maclellan & Co. Ltd., republication by Dover, 1973.
- [Bai86] Bain, I. 1986. Celtic Knotwork, Constable and Company Ltd., republication by Sterling Publishing Company, 1992.
- [Bax04] Baxter, W. 2004. Physically-Based Modeling Techniques for Interactive Digital Painting, PhD thesis, University of North Carolina at Chapel Hill.
- [Bon99] Bonabeau, E., Dorigo M., and Theraulz G. 1999. Swarm Intelligence: from Natural to Artificial Systems, Santa Fe Institute, Oxford University Press.
- [Bro04] Brown, K. and Clark R. 2004. The Lindisfarne gospels and two other 8th century Anglo-Saxon/Insular manuscripts: pigment identification by Raman spectroscopy, Journal of Raman Spectroscopy, 35, 4-12.
- [Bro05] Browne, C. 2005. Chaos and Graphics: Cantor knots, *Computers and Graphics*, 29(6), December, 998-1003. Browne, C. 2006. Chaos and Graphics: Wild knots, *Computers and Graphics*, 30(6), December, 1027-1032.
- [Chu05] Chu, N. and Tai, C. 2005. MoXi: Real-Time Ink Dispersion in Absorbent Paper, Proceedings of ACM SIGGRAPH, 504-511.
- [Coo00] Cooper, D. 2000. Early Manuscripts at Oxford University, <http://image.ox.ac.uk/>, Oxford digital library.
- [Cra06] Cramp, R. et al. 2006. Corpus of Anglo-Saxon Stone Sculpture, Durham University and The British Academy, <http://www.dur.ac.uk/corpus>.

- [Cro93] Cromwell, P. 1993. Celtic knotwork: Mathematical Art, *The Mathematical Intelligencer*, 15, 1, 36-47.
- [Dre00] Drewes, F. and Klempien-Hinrichs, R. 2000. Picking Knots from Trees: The Syntactic Structure of Celtic Knotwork, *Lecture Notes in Computer Science*, 1889, 77-106.
- [Doy08] Doyle, R. 2008. Computer Generated Celtic Canvas, PhD Thesis, Advisor: SK Semwal, Department of Computer Science, University of Colorado, Colorado Springs, pp. 1-230.
- [Far97] Farin, G. 1997. Curves and Surfaces for CAGD: A Practical Guide, 4th Ed., Morgan Kaufmann and Academic Press, 118-121.
- [Fle95] Fleischer, K. 1995. A Multiple-Mechanism Developmental Model for Defining Self-Organizing Geometric Structures, PhD thesis, California Institute of Technology.
- [Fre03] Freeman, W. 2003. Learning Style Translation for the Lines of a Drawing, *ACM Transactions on Computer Graphics*, 22, 1, 33-46.
- [Fun07] Fung, K. 2007. Celtic Knot Generator, BSc (Hons) thesis, University of Bath.
- [Ger05] Gerber, E. 2005. A Stochastic Model for the Spatial Structure of Annular Patterns of Variability and the North Atlantic Oscillation, *Journal of Climate*, 18, 2102-2118.
- [Ger90] Gerdes, P. 1990. On ethnomathematical research and symmetry, *Symmetry; Culture and Science*, 1, 2, 154-170.
- [Gra98] Alfred Gray, Elsa Abbena, and Simon Salamon (1998), *Modern Differential Geometry of Curves and Surfaces with Mathematica*, 1998, Chapman and Hall.
- [Gla99] Glassner, A. 1999a. Andrew Glassner's notebook: Celtic Knotwork, Part I, *IEEE Computer Graphics and Applications*, 19, 5, 78-84.
- [Yev02] Yevin, I. 2002. Criticality of the Brain and Criticality of the Art, *Proceedings of the Fourth International Conference on Complex Systems*, paper #209.
- [Har76] Hardy, J. and de Pazzis O. 1976. Molecular dynamics of a classical lattice gas: Transport properties and time correlation functions, *Physical Review A*, 1949-1961.
- [Hen74] Henry, F. 1974. *The Book of Kells: Reproductions from the Manuscript in Trinity College Dublin*, Thames and Hudson.
- [Ise06] Isenberg, T., Neumann, P., Carpendale, S., Sousa, M., and Jorge, J. 2006. Non-photorealistic rendering in context: an observational study, *Proceedings of the 4th International symposium on Non-photorealistic animation and rendering*, 115-126.
- [Itt04] Itti, L., Dhavale, N., and Pighin, F. 2004. New realistic avatar and head animation using a neurobiological model of visual attention, *Proceedings of SPIE 48th Annual International Symposium on Optical Science and Technology*.
- [Jab95] Jablan, S. 1995. Mirror generated curves, Symmetry, Culture, and Science, quarterly publication for the Interdisciplinary Study of Symmetry, 6, 2, 275-278.
- [Kap03] Kaplan, M. and Cohen E. 2003. Computer Generated Celtic Design, *Eurographics Symposium on Rendering*.
- [Lar07] Larboulette, C. 2007. Celtic Knot Colorization based on Color Harmony Principles, *Computational Aesthetics*.
- [Mer08] Meraj, Z. 2008, Wyvil, B, Isenberg, T, Gooch, A, and Guy, R. Mimicking Hand Drawn Pencil Lines, *Computational Aesthetics*.
- [Mel96] Malevanets, A. and Kapral R. 1996. Links, Knots, and Knotted Labyrinths in Bistable Systems, *Physical Review Letters*, 77, 4.
- [Mee91] Meehan, A. 1991. *Celtic Design: Knotwork: The Secret Methods of the Scribes*, Thames and Hudson.
- [Mee03] Meehan, A. 2003. *Celtic Knots: Mastering the Traditional Patterns, A Step by Step Guide*, Thames & Hudson.
- [Mee94] Meehan, B. 1994. *The Book of Kells: An Illustrated Introduction to the Manuscript in Trinity College Dublin*, Thames and Hudson.
- [Mer97] Mercat, C. 1997. Les Entrelacs des Enluminures Celts, *Pour La Science*, 15.
- [Mit09] Mitchell, M. 2009. *Complexity – A Guided Tour*, Oxford Press, 1-368, ISBN10:0195124413.
- [Mue97] Muller-Hannemann, M. (1997). *Quadrilateral Mesh Generation in Computer-Aided Design*, PhD thesis, Technische Universitat Berlin.
- [Per02] Perlin, K. 2002. Improving Noise, *Computer Graphics*, 35, 3, 681-682.
- [Pru90] Prusinkiewicz, P. and Lindenmayer, A. 1990. *The Algorithmic Beauty of Plants*, Springer.
- [Qui07] Quinn, E. 2007. Modern techniques seek some secrets from ancient Irish manuscript, *International Herald Tribune*, May 28.
- [San05] Santella, A. 2005. *The Art of Seeing: Visual Perception in Design and Evaluation of non-Photorealistic Rendering*, PhD thesis, Rutgers.
- [Sim94] Sims, K. 1994. Evolving Virtual Creatures, *Proceedings of ACM SIGGRAPH*, 14-22
- [Sou03] Sousa, M. and Prusinkiewicz, P. 2003. A few good lines: Suggestive drawing of 3D models, *Proceedings of Eurographics: Computer Graphics Forum*, 22, 3, 381-390.

Haptic Technique for Simulating Multiple Density Materials and Material Removal

Tales Nereu Bogoni^{1,2,3}, Márcio Sarroglia Pinho^{1,2}

¹PontificalCatholicUniversityof Rio Grande do Sul
Porto Alegre – RS – Brazil

²INCT-MACC

Petrópolis – RJ – Brazil

³Mato Grosso StateUniversity
Colider – MT –Brazil

tales@unemat.br, pinho@puccrs.br

ABSTRACT

The term *haptic* refers to the tactile sensation perceived by the user when he is manipulating objects in a Virtual Environment. These sensations are extremely important in Virtual Reality simulators for training medical and dental procedures with the goal of increasing the level of motor skills. Many tasks trained, such as teeth and bones drilling, are held in rigid bodies and require parts of the virtual models to be removed. One way to manipulate these objects is by using voxel-based models, which divide the object into smaller parts that can be removed by the haptic device. This paper deals with a new method for haptic rendering with voxel-based rigid bodies that may be composed of various materials with different densities. The method describes the process for obtaining and storing the virtual objects. In addition it details the techniques used for collision detection, calculating the force feedback and removing parts of objects during the drilling process. The paper ends with an experiment that shows that the method provides stability, supports removal of voxels and different material densities properly.

Keywords

Virtual Reality; Haptic Rendering; Voxel-based Material Removal

1. INTRODUCTION

Virtual Reality (VR) has been increasingly used for training in various areas. One of these fields is health, which uses simulators to train surgeons or dentists, with the aim of increasing their motor skills before they have direct contact with patients. In order to do this, simulators must provide a high degree of realism, both visual and tactile.

For simulators to produce tactile sensation, they must contain haptic devices. These devices reproduce in the real environment the effects of the interactions happening in the virtual environment, causing reactions of the equipment that is being manipulated by the user. These reactions are called *haptic feedback*. The strategy used for calculating the force feedback is called *haptic rendering* and seeks to represent the resistance of the materials that compose the virtual objects during the interaction of a virtual instrument, which is being manipulated by the user with the other virtual objects.

One of the tasks that make use of haptic devices is sculpting rigid virtual objects. During this process,

the operator has to remove part of the volume of the object in order to change its original shape. In the healthcare area, the process of sculpting may be applied in systems that seek to train bones scraping and drilling, and also in dentistry to prepare teeth for filling or for endodontic treatments. Bones and teeth are composed of various types of tissues with different degrees of resistance to perforation. Thus, it is necessary to use virtual models able to represent objects made of multiple materials. In general, these models are made of voxels, even though there are approaches that use polygonal models. Among the various methods that perform haptic rendering using volumetric models, a great part does not deal with changes in the structure of the object such as voxel removal by drilling or sculpting, and are used for exploring the models only.

Considering this background, the aim of this research is to present a method that can be used for sculpting rigid objects composed of voxels with different degrees of hardness, so that they can be used to represent bones and teeth in medical and dentistry training systems. The method presented in this paper is able to keep the stability of the haptic device and represent realistically the force feedback that occurs during the interaction between the tool manipulated by the user and the object being sculpted.

In addition to this introduction, the paper is divided into four sections. Section III presents the commonly adopted strategies for generating force in haptic devices, and also the basic concepts related to haptic

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

rendering and similar research from a bibliographic review in healthcare articles that mention haptic devices and voxelized objects. Then, Section IV describes the method used to perform the experiment, including the modeling and the representation of the object to be sculpted and the tool, how collision detection and haptic rendering are performed, and finally the removal of voxels from the volume. Section V describes an experiment used for evaluating the method proposed in this paper. Finally, Section VI presents the conclusions obtained in this study and suggestions for future research.

2. STRATEGIES FOR GENERATING FORCE FEEDBACK IN HAPTIC INTERFACES

For the development of this research, a bibliographic review was carried out in order to identify the main strategies for obtaining the objects that compose the virtual environment, the data structures used for storing these objects, the methods applied to collision detection and the strategies adopted for calculating the force feedback that is sent to the haptic device. Finally, techniques developed for removing the object parts are described, all of them applied to systems related to the healthcare area.

Ways of obtaining the objects

Within the healthcare area, the objects represent parts of the human body and are obtained basically in 3 ways: tomographic images, magnetic resonance and simplified geometric models. In the first two cases, sets of segmented images are used to identify the different types of tissue and set the values of the tissue density to the voxels, based on the intensity of the voxels identified in the segmented images. In the last case, geometric models are converted into volumetric objects with density values arbitrarily defined for each type of material.

Data storage structures

Both in the area of medicine [5, 10, 13] and in dentistry [1, 11, 15], most systems make use of three-dimensional matrices for storing the object to be sculpted. This is a common approach, because each voxel can store the density of the represented tissue (used for calculating the force feedback in the haptic device) regardless of the neighbor structures. Besides, because they are obtained from tomography and magnetic resonance, the representation through voxels is more natural.

In addition to the objects to be sculpted, the systems have virtual tools that are manipulated by the user to sculpt the object. Such tools may be represented in polygonal [2, 5, 13], volumetric [9, 22], or hybrid [1, 10, 15] ways.

Methods for collision detection

Bounding boxes are used with geometrically modeled tools, but when volumetrically modeled tools are used, collision detection is accomplished through an occupancy map [12]. In this structure, the matrices that represent the object and the tool are superposed, which makes it possible to identify the voxels with collision.

With tools modeled in a hybrid way, the tool is wrapped around bounding boxes, and when it collides with the object's bounding box, the cutting part of the tool (drill bit) is transformed into voxels and superposed to the voxels of the object.

Strategies for calculating force feedback

After the collision detection, it is necessary to simulate the force vectors resultant from the contact between the tool and the object being sculpted and send them to the haptic device to perform the haptic rendering, in order to make the user exert more or less force to move, penetrate or break the object. This simulation uses data intrinsic to the objects in the virtual environment, such as hardness, texture and friction coefficients, in addition to vectors related to the movement performed by the haptic device and the force exerted by the user. Due to its complexity, maintaining a suitable refresh rate in this simulation is the greatest challenge faced by haptic systems. In order to allow an interaction without vibrations or abrupt movements for the user, the devices must be refreshed about 1000 times per second (1 KHz)[4], which makes it necessary to perform collision detection and force feedback calculation in less than 1 ms.

In the real world, when there is an interaction with a rigid object by using a tool, this tool remains in the surface of the object when the contact occurs. On the other hand, when a haptic device is used, the *Haptic Interface Point* (HIP), which is responsible for telling the system which point is being manipulated in the device, can penetrate the virtual object between two processing cycles, depending on the speed of the movement performed by the user. In those cases, the haptic device must apply a force that acts against the movement performed by the user, making the HIP return to the surface of the virtual device. The point that defines the feedback position is called proxy or god-object [23].

Determining the correct position of the proxy is a hard task, and is a decisive issue to make the system able to keep the stability of the haptic rendering. During the execution of each haptic rendering loop, it is necessary to identify the position of the HIP, find the new position of the proxy and calculate the force that will be sent to the haptic device.

Figure 1 shows the haptic rendering process. It is possible to observe that the HIP and the proxy have

the same coordinates while there is no collision with the object, which can be noticed in $t-3$, $t-2$, $t-1$. In time t the HIP penetrates the object and the proxy is kept on its surface. Then, in time $t+1$, a new position is calculated for the proxy when the HIP is moved to a new position inside the object, with the aim of keeping it on the surface of the object, but following the HIP movement.

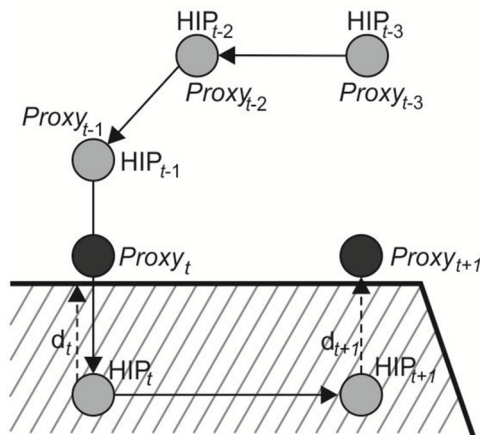


Figure 1–Haptic Rendering (Adapted from [6])

As to polygonal models, the normal vectors of the planes that compose the object are used to indicate the position of the proxy [17, 23].

With regard to volumetric objects, however, these normal vectors do not exist explicitly. For these cases, the literature presents two approaches. In the first one, these normal vectors are calculated from the coordinates of the voxels in the surface of the object [18], and then the same techniques described for Figure 1 are used. Another solution is to apply methods that determine the new proxy position directly from the voxels, using the previous proxy coordinates and the current HIP, as presented by Vlasov et al. [19], who use a Ray Casting algorithm between the previous position of the proxy and the current position of the HIP to calculate the new position of the proxy, which will be close to the surface of the object. In this technique, the first step is to cast a ray from the current position of the proxy to the HIP, creating a list of voxels that are crossed by the ray. The closest voxel to the current proxy is defined as the voxel in which the collision between the object and the tool occurred. Then, the path of the ray is travelled with small steps, smaller than the size of a voxel, until it reaches the voxel of the collision. After that, the last position visited, before reaching it, is defined as the new proxy.

After defining the proxy, it is necessary to calculate the force vector and send it to the haptic device. This vector is normally calculated with Hooke's Law $\mathbf{F} = k\mathbf{x}$, where k is a stiffness constant of the material and \mathbf{x} is the vector between the position of proxy and the HIP [6]. In order to illustrate this calculation,

Figure 1 presents the vector (d_t) as the vector between the HIP and the proxy, which must be multiplied by the stiffness constant of the material being simulated. In $t+1$ a new force vector is calculated by using the new positions of the HIP and the proxy. This process is repeated after each loop of the haptic rendering.

Content removal techniques

The haptic systems may be used to remove parts of the material that form the objects based on the action of the tool. Depending on the density properties of the material and the force with which the operator uses the haptic device, the removal can be done faster or slower [12]. In the most common approach, the voxel density is gradually reduced to zero, when the voxel is eliminated. In general, rotatory tools with fixed [22] or variable [19] speed are used.

3. PROPOSED METHOD

Our method for haptic rendering proposed has been developed with the aim of using volumetric objects with multiple materials to represent rigid structures of the human body, such as teeth and bones, in procedures that involve drilling and scraping with tools that have a drill bit.

The next sections describe the techniques applied in this method to represent the objects to be sculpted and the tools manipulated by the user, the collision detection process, the algorithm to choose the proxy position, and the strategies to generate force feedback and to perform the volume removal.

Representing the objects to be sculpted

The objects are stored in a volumetric way using a three-dimensional matrix, in which each entry represents a voxel. The voxels contain an attribute that indicates its density, determining the type of material it is representing. The density aims at informing the graphic renderer if the voxel is visible or not, and the haptic renderer about the amount of material remaining to be removed. When the voxel has a positive density, it is used in the graphic rendering and haptic rendering algorithms; when it is zero, the voxel is ignored.

Another characteristic in voxels concerns their position in relation to their neighbors. The possible classes are: **external** voxel, when it presents null density; **internal** voxel, when it has density and has no external neighbors; and **boundary** voxel, when it has a positive density and at least one external neighbor. The classification of external and internal voxels is simple, by simply checking its density. For the classification of boundary voxels, it is necessary to analyze their 26 neighboring voxels, which is called 26-neighbor (Figure 2a) of each voxel. If the voxels in this neighboring are not all of the same type (internal or external), the voxel is classified as a boundary one.

With this information, the occupancy map of the object is obtained, which is illustrated two-dimensionally in Figure 2b, indicating the external (0), internal (2) and boundary (1) voxels.

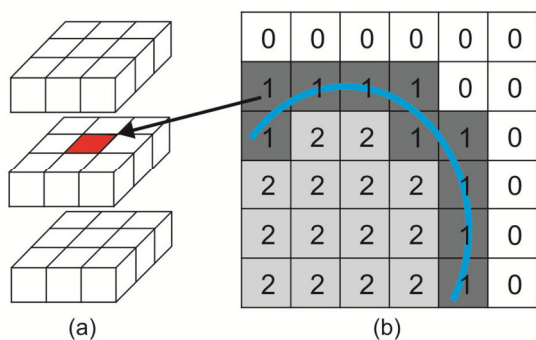


Figure 2- 26 Neighboring (a) and occupancy map (b) (Adapted from [12])

Other relevant information about voxels refers to its coordinates. Two coordinate systems are used, a local and a global one. In the local system, the coordinates represent the position of the voxel inside the 3D matrix, which is used to facilitate the localization of the voxel's neighbors. In the global one, the voxel coordinates are stored inside the coordinate system of the haptic device, which are used in the haptic rendering algorithm for collision detection and force feedback calculation. In order to complete the object modeling, a bounding box that delimitates the volume occupied by the object's voxels is stored.

Representing the tools

The tools are modeled in a hybrid way, with a geometric model for the body of the tool and a volumetric model for the part used for drilling the volume (drill bit). Figure 3 shows some possible drill bit models, but other models are possible.

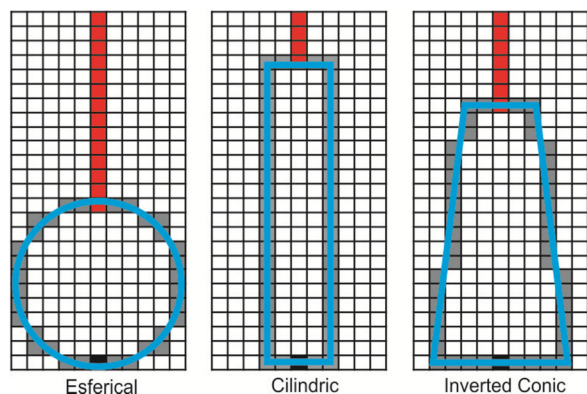


Figure 3-Drill bit models

For the volumetric definition of the drill bits, two types of voxels are used, one in the cutting part of the drill bit, which is responsible for removing the material of the object to be sculpted (represented in gray in Figure 3), and another one (in red) that

represents the drill bit handle and is used for collision detection only.

For the haptic rendering algorithms to work correctly, the size of the voxels used to represent the drill bit of the tool must be the same used for the object to be sculpted. The displacement of the tool in the virtual environment happens through the manipulation of the haptic device that controls the HIP, which works as a pivot of the tool. This pivot is represented in Figure 3 and Figure 4 by a black voxel. Based on the pivot, the tool's voxels will be superposed in the occupancy map of the object in order to check the existence of collision and perform the removal of the object's voxels. In the example illustrated in Figure 4a, the drill bit is inside the object's occupancy map, but without collision, while in Figure 4b it is colliding with the boundary. More details on the process of collision detection will be presented next.

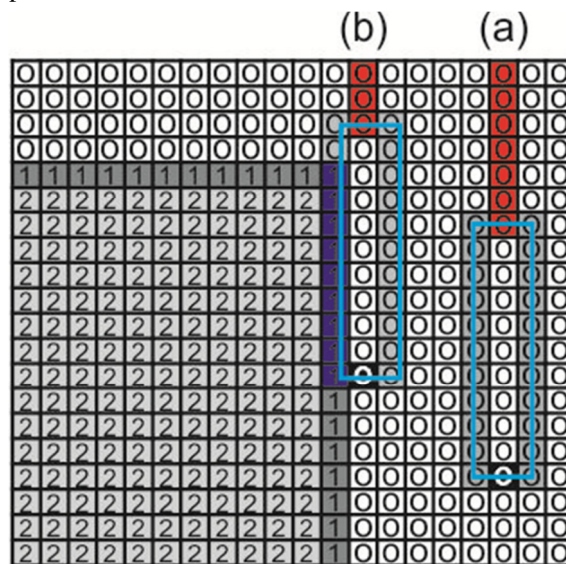


Figure 4-Occupancy map with superposed drill bit

Collision detection

Collision detection is carried out by checking if the drill bit of the tool is touching the border of the object to be sculpted. The algorithm used for this is an adaptation of the algorithm proposed by McNeely et al. [12], which uses an occupancy map to check the existence of collision.

In the approach proposed in this paper, collision detection is performed in two stages. In the first one, we check if the tool's pivot point is inside the bounding box of the object. If it is, the tool's matrix of voxels is superposed to the object's voxels, so that the collision verification is carried out with each voxel of the tool. The algorithm considers that there is a collision when a tool's voxel occupies the same space of an object's voxel, with non-null density, that is, with internal or boundary voxels.

Figure 5 shows the process of collision detection during the tool's movement. In picture (a) the tool's pivot is outside the object's bounding box and the other voxels of the tool are not considered. In pictures (b) and (c) the pivot penetrates the object's bounding box and the tool's voxels are superposed to the object's voxels without any collision. In picture (d) there is a collision between the tool's voxels and the object's boundary.

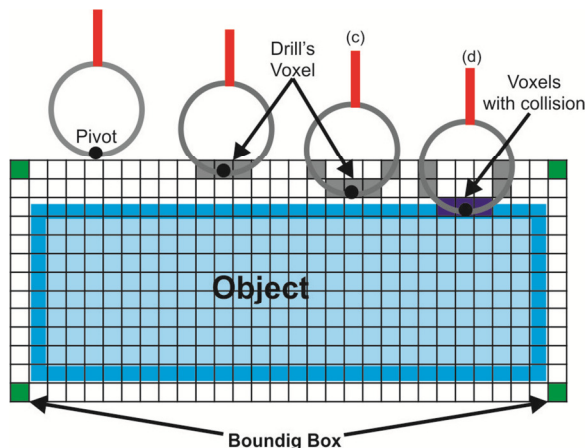


Figure 5–Collision Detection

Choosing the position of the proxy

The method presented in this paper is based on a technique presented by Vlasov et al. [19], which uses a Ray Casting algorithm to determine the proxy's position. The calculation of the proxy's position begins when a collision between the tool and the bounding box that covers the object being sculpted is detected. In this moment, the occupancy map is used to create a list with all the voxels of the object classified as external and that belong to the 26-neighbor of the voxel that contains the proxy. Among these voxels, the closest voxel to the current position of the HIP is found, which will be the voxel containing the new proxy. Figure 6 illustrates the process of choosing the voxel that will contain the proxy. In $t-1$ the current voxel containing the proxy (in gray) and the external neighbors of this voxel (in yellow) are presented. In $time t$ the HIP is moved and we check which of the neighbor voxels is closer to the HIP, defining it as the voxel that will contain the proxy.

At first, the position chosen for the proxy is the center of the closest voxel to the HIP, which can be observed in Figure 6. However, depending on the size of this voxel, the displacement to move the HIP becomes too large and may cause instability in the system. In order to avoid this, a ray from the center of this voxel is casted up to the current position of the HIP. Then this ray is sampled in small points (not exceeding 0,1mm [21]) and these points (Figure 7) are visited from the center of the chosen voxel to the position of the HIP, until a bordering voxel is

reached. When it happens, the last visited point is chosen as the new proxy's adjusted position.

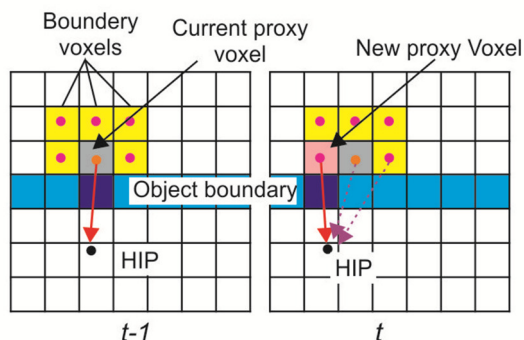


Figure 6 – Determining the voxel that will contain the proxy

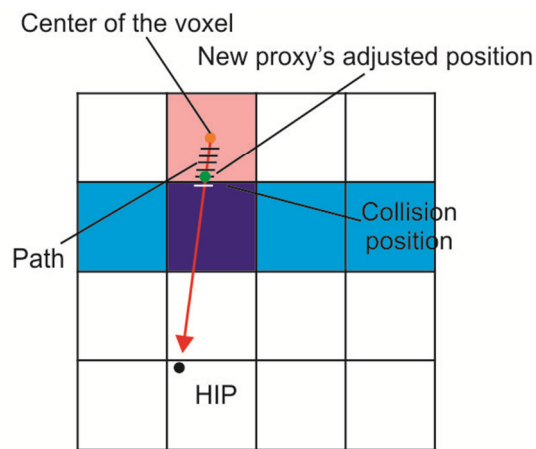


Figure 7–Adjustment of the proxy's position

Force Feedback

As it has already been described above, the virtual tool always follows the proxy and should not penetrate the bordering or internal voxels of the object being sculpted. Therefore, it will navigate only where the occupancy map indicates the existence of external voxels. The HIP, on the other hand, can penetrate the object, and its distance in relation to the proxy is used to indicate the direction and magnitude of the force the user is exerting on the haptic device. Depending on the displacement of the HIP, the tool can slip through the object's boundary, as long as there are no obstacles preventing this movement. Figure 8 illustrates the movement of the HIP and the proxy, where (a) shows the HIP and the proxy together, and (b) shows the proxy in the boundary and the HIP inside the object. When the HIP penetrates the object, it is possible to calculate the force vector that will be sent to the haptic device, which will go in the same direction but in an opposite way to the vector that connects the proxy and the HIP. The magnitude of the vector is given by the distance between the proxy and the HIP.

To determine the resulting force (\vec{F}) that will be sent to the haptic device, two vectors are used, as shown in Equation (1). The \vec{F}_R vector represents the resistance of the material that composes the boundary of the object in collision, and the \vec{F}_F , with lower intensity, indicates the frictional force occurring between the voxels of the tool and the boundary of the object.

$$\vec{F} = \vec{F}_R + \vec{F}_F \quad (1)$$

The resistance vector (\vec{F}_R) is calculated with Equation (2), where, P_{Proxy} is the position of the proxy in the coordinates of the haptic device, P_{HIP} is the current position of the haptic device's pointer, and K is the stiffness constant of the material of the voxel where the proxy is located.

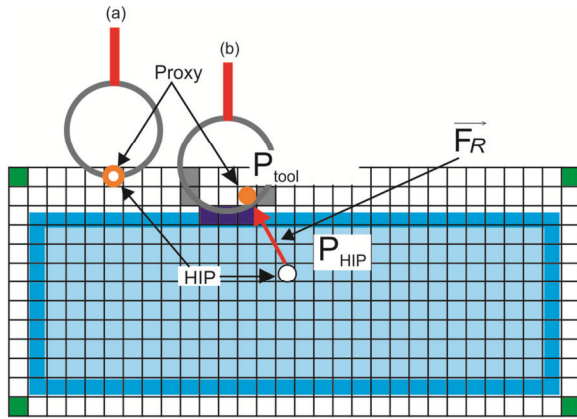


Figure 8 - Components of the Force Feedback Calculation

$$\vec{F}_R = (P_{Proxy} - P_{HIP}) * K \quad (2)$$

Besides the distance between the proxy and the HIP, it is necessary to take into consideration the number of voxels in collision between the tool and the object. This generates a frictional force, \vec{F}_F , which is calculated from the average of the distances between the voxels in collision and the center of the tool. Equation (3) presents the formula of \vec{F}_F , where n is the amount of the tool's voxels in collision with the object, P_{Center} is the position of the center of the tool in the device's coordinates, $P_{Collision(i)}$ is the position of each collision point in the device's coordinates, and K_T is the frictional constant of the tool's material.

$$\vec{F}_F = \frac{\sum_{i=0}^n P_{Center} - P_{Collision(i)}}{n} * K_T \quad (3)$$

Figure 9 shows two examples of frictional force vector calculation. On the left, the contact occurs with three voxels from the bottom, which makes the force vector point up wards, while on the right the collision occurs both at the bottom and on the side of

the tool, which makes the force vector point on a diagonal line.

Volume Removal

The collisions between the tool's cutting parts and the object's surface cause the removal of voxels and the appearance of holes in the object. Consequently, the shape of the object changes with the elimination of some boundary voxels, which turns internal voxels into boundary ones.

The method applied in this paper was inspired by the material removal model proposed by Wang et al. [20], which uses rotary tools for dental drilling. In their method, the amount of material to be removed is calculated based on the tangential velocity of the tool and its displacement. Besides, the authors point out that the amount of material removed from the object will not be greater even if excessive force is exerted during the drilling process, due to the physical limit of the haptic device being used.

Vector of distance between the center of the drill bit and the voxels with collision

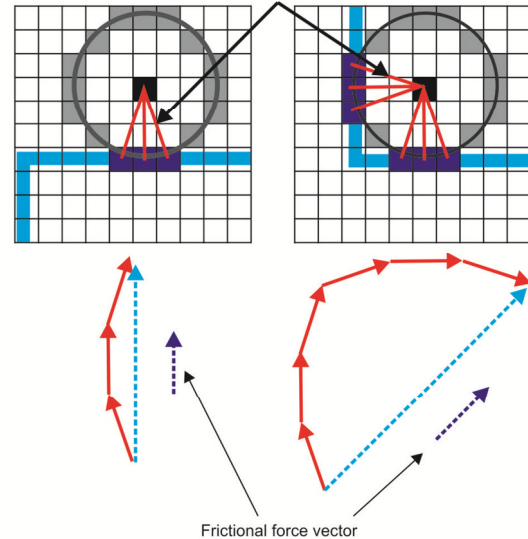


Figure 9 – Voxels with collision for Force Feedback Calculation through the friction between the drill bit and the object's boundary

In the method presented in this paper, a voxel is removed when its density reaches zero. The reduction in voxel density occurs gradually, taking the force exerted by the user on the haptic device into consideration and respecting the device's limit of force feedback, without using the tangential velocity of the tool.

In order to quantify the amount of material to be removed, a wear coefficient (α) is used, which is calculated with Equation 4. For this calculation it is necessary to determine the maximum force feedback value supported by the haptic device (d_{max}), and the distance between the proxy and the HIP (Δd), indicating when the maximum wear of the material will be achieved. Thus, if the user applies a little

force, a small amount of material will be removed, and if the user applies more force than the limit supported by the haptic device, only the amount of material allowed by the system will be removed.

$$\alpha = \begin{cases} 1 & \text{se } \Delta d \geq d_{\max} \\ \frac{\Delta d}{d_{\max}} & \text{se } \Delta d < d_{\max} \end{cases} \quad (4)$$

Figure 10 illustrates how the wear coefficient calculation is done. In the left picture, the HIP is inside the force limit accepted by the haptic device (d_{\max}), thus α is calculated. In the picture on the right, the HIP exceeds the limit, so α assumes the maximum value allowed and the exceeding force applied by the user is ignored.

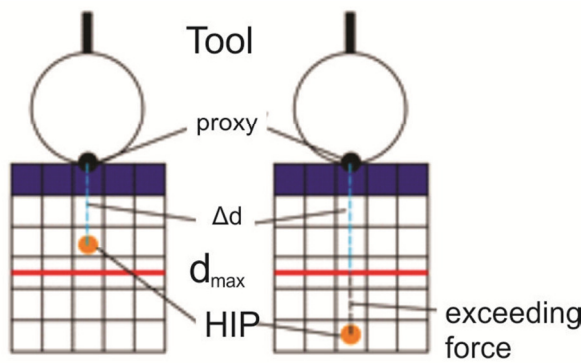


Figure 10 – Demonstration of the calculation of the multiplication factor for voxel removal.

Besides the wear coefficient, a constant representing the material that constitutes the drill bit is applied, which may scrape the object with more or less intensity. The final calculation that will define the new density of the object's voxels is performed on haptics data every 1ms by using Equation (5), where D is the current density of the voxel, K_{Tool} is the constant that represents the material of the tool's drill bit, and α is the wear coefficient. As the graphics rendering is slower than the device update, the new data are sent to the graphic renderer at every 25ms.

$$D' = D - k_{Tool} * \alpha \quad (5)$$

When the density of a voxel gets null, the voxel is eliminated from the haptic and graphic rendering and the object's boundary must be redefined. In order to do this, it is necessary to reclassify the neighbors of the removed voxel. In this process, the internal voxels of the 26-neighbor of the removed voxel are selected and each of them is reclassified. Figure 11, on the left, shows the collision between the voxels of the tool and of the object, and on the right, it demonstrates the boundary after the removal of the voxels that were in collision.

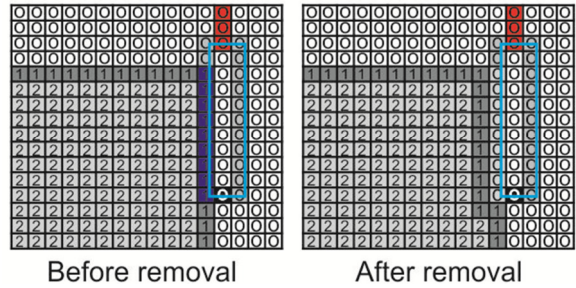


Figure 11–Update boundary

4. Experiment

The method proposed in this paper has been implemented on the C++ programming language using Visual Studio 2010, the OpenGL graphic API and the Chai3d [7] library to control the haptic device. In order to move the tool, the system uses a Novint Falcon Controller [8] device, with 3DOF for tracking and force feedback and 8,9N of force feedback in each axis. We used an Intel Core i5 3.1Ghz processor desktop computer, with 4GB of memory and an AMD Radeon HD 7570 video controller. The system was developed with the use of two *threads*, one of them responsible for the haptic rendering with 1 KHz minimum update, and the other one responsible for the graphic rendering only, which works on about 30 Hz. The tasks of tracking and updating are accomplished inside the haptic renderer, in addition to tasks of collision detection, force feedback calculation and voxel removal.

As to the experiments, we used an object composed by three materials created from polygonal models and converted to voxels with a 128x128x128 voxel resolution, with the software BinVox [3]. Figure 12 shows the stacked block-shaped materials forming a cube. Each block is 10 mm long and wide and 3.33 mm high, so that the cube formed by the stacking of these materials measures 10 mm in each edge. Therefore, each voxel measures 0,078 mm (10 mm / 128 voxels) in each dimension. The density of the voxels is calculated based on the time required to move the tool during the process of drilling, as it will be presented next.

The tool used in the experiment has a polygonal model, used by the graphing renderer only, and a volumetric model, which represents a 0.5mm spherical drill bit.

We developed two kinds of tests. The first one aimed to check the stability of the method by measuring the refresh rate of the device. It also checked whether or not the force feedback is consistent with the movement of the tool. In the second test, the purpose was to verify if the method is able to simulate materials with different degrees of resistance to perforation.

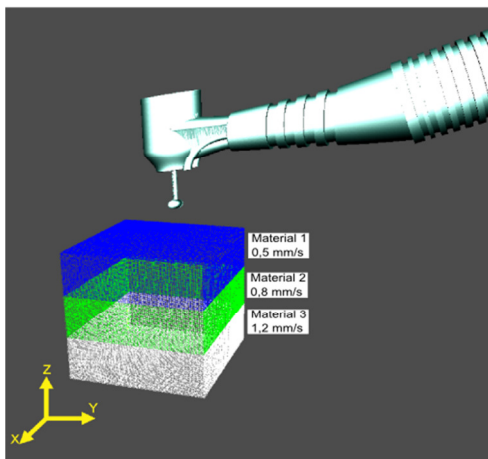


Figure 12 – Screenshot of the models used in the experiment

To evaluate the refresh rate of the device, data related to the use of the system were collected every 25ms, while the user performed the **displacement**, **exploration** and **perforation** procedures. Displacement occurs when there is no contact between the tool and the object being sculpted. Exploration happens when the tool slips through the object without altering the density of the voxels. Perforation occurs when the tool reduces the density of the voxels and eliminates it. In order to activate the perforation mode it is necessary to keep pushing a button on the haptic device.

The graph in Figure 13 shows the data collected for the refresh rate of the haptic device. It can be noticed that in the displacement task (A), the refresh rate is high, as only the collision of the object’s bounding box with the HIP is detected, and the device is free to move. In the exploration task, the refresh rate reduces to about 62 KHz, as the force feedback calculations and the collision detection with the voxels of the tool are performed. In the perforation task, the refresh rate is about 50 KHz, due to the number of voxels of the tool involved in the force feedback calculation as well as the data update inside the object’ matrix for reclassifying the voxels to form the new boundary.

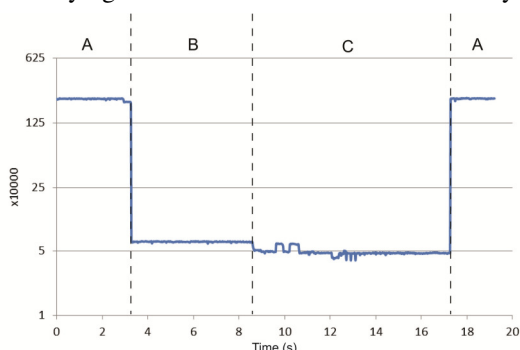


Figure 13 – Refresh rate of the haptic device, in 10 KHz

As to the displacement of the tool and the force feedback, it is possible to see in the graph in Figure 14 that during the navigation task (A) the magnitude of the displacement is great in all the axes and the force feedback is null. When the exploration task is performed (B), the axis with higher force feedback indicates the existence of collision and, as a result, less magnitude of the displacement. In this example, the movement is in the upper part of the object, with higher force feedback and less movement in the Z-axis, which indicates that the user is trying to penetrate the object downwards and sliding on the surface. Finally, in the perforation task (C), the greater the magnitude of the movement, the higher is the force feedback in the axis. In other words, the more the user moves towards a direction, the more the haptic device works against the movement, with the goal of keeping the tool in external voxels.

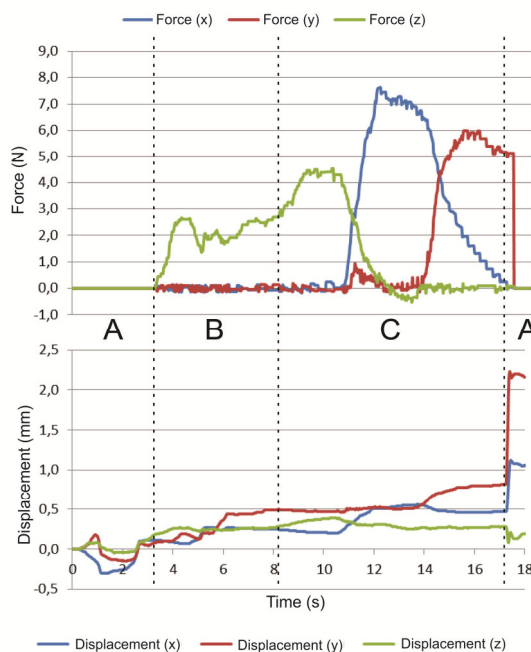


Figure 14 – Displacement of the tool and force feedback

The second test was developed with the goal of checking the materials’ resistance to perforation. For that purpose, it is necessary to simulate objects with different densities. To determine density, the tool’s displacement velocity is measured in a perforation task during the displacement of the tool in the Z-axis, passing it through the three materials, as in the research by Wang et al. [20] and Arbabtafti et al [2].

For this example, we simulated materials whose perforation velocities are 1.2mm/s, 0.8mm/s and 0.5mm/s. These values respectively represent the perforation times of dentin [20], bone [11] and enamel [20], common materials in dentistry procedures. Equation (6) was used to calculate density, where D is the density of the voxel, APS is

the amount of volume removal updates performed in 1 second, $VPMM$ is the amount of voxels that fit into 1.0 mm, and RV is the removal velocity in mm/s. As the size of the voxel is 0.078mm in this example and the update occurs every 25ms, $VPMM = 12.8$ voxels/mm and $APS=40$ updates/s. By applying Equation (6), the density values associated with each material were 2.6, 3.9 and 6.24.

$$D = \left(\frac{APS}{VPMM} \right) \div RV \quad (6)$$

The graph in Figure 15 presents the results obtained in this experiment. It is possible to notice that by using the density values calculated and applying the maximum allowed force of density reduction, the displacement of the tool during the perforation process is coherent with the drilling velocity calculated.

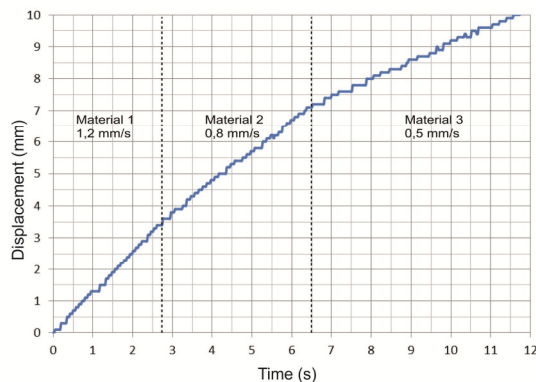


Figure 15 – Displacement of the haptic device during perforation

5. Conclusions and Future Research

The use of the method proposed in the paper makes it possible to model both simple rigid objects with homogeneous density, such as metal or wood blocks, and complex objects formed by materials with different degrees of hardness, such as teeth and bones. The method is independent on the way the data are acquired, as long as they can be stored in a three-dimensional matrix. Thus, it is possible to use data obtained from computer tomography, magnetic resonance or polygonal models converted into voxels.

Considering that the system allows the user to perform displacement, exploration and perforation, the results of the tests demonstrate that the force feedback for the haptic device was kept stable during all the tasks, and it was always updated above 1 KHz, which caused neither undesired displacement in the tool nor vibration in the user's hand. This allowed the user to feel the collisions between the tool and the object and to remove voxels properly.

In addition, the process of drilling and material removal has been effective in correctly removing the voxels and reconstructing the object's boundary in every removal, adhering to the estimated time for the voxel removal based on the density of each material.

Future research can make use of devices with 6 DOF, which allow the rotation of the tool, besides including the tangential velocity control of the tool. Finally, the method presented in this paper can be applied to implement a simulator for training professionals in the area of dentistry with a setup composed by specific hardware that allows the rotation of the tool and controls the tangential velocity of the drill bits by customizing the Novint Falcon Controller haptic device.

6. Acknowledgements

Our research is funded by the National Institute of Science and Technology in Medicine Assisted by Scientific Computing (Grant CNPq 181813/2010-6 and FAPERJ E-26/170.030/2008).

7. References

- [1] Acosta, E.; Liu, A. "Real-time Volumetric Haptic and Visual Burrhole Simulation." IEEE Virtual Reality Conference, 2007. VR '07. p.247-250, 2007. IEEE.
- [2] Arbabtafi, M.; Moghaddam, M.; Nahvi, A.; Mahvash, M.; Rahimi, A. "Haptic and visual rendering of virtual bone surgery: A physically realistic voxel-based approach." IEEE International Workshop on Haptic Audio visual Environments and Games, 2008. HAVE 2008. p.30-35, 2008. IEEE.
- [3] binvox 3D mesh voxelizer. www.cs.princeton.edu/~min/binvox
- [4] Cavusoglu, M. C.; David, F.; Frank, T. "A critical study of the mechanical and electrical properties of the phantom haptic interface and improvements for highperformance control." Presence: Teleoperators & Virtual Environments 11.6 (2002): 555-568.
- [5] Eriksson, M.; Flemmer, H.; Wikander, J. "A haptic and virtual reality skull bone surgery simulator." Proceedings of World Haptics, 2005.
- [6] Ho, C-H, Basdogan, C.; Srinivasan, M.A. "Efficient point-based rendering techniques for haptic display of virtual objects." Presence 8.5 (1999): 477-491.
- [7] <http://www.chai3d.org>
- [8] <http://www.novint.com>
- [9] Kimin Kim; Ye-Seul Park; Jinah Park. "Volume-based haptic model for bone-drilling." International Conference on Control, Automation and Systems, 2008. ICCAS 2008. p.255-259, 2008. IEEE.
- [10] Ki-Uk Kyung; Dong-Soo Kwon; Sung-Min Kwon; Heung Sik Kang; Jong Beom Ra. "Force feedback for a spine biopsy simulator with volume graphic model." 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2001. Proceedings. v. 3, p.1732-1737 vol.3, 2001. IEEE.
- [11] Kusumoto, N.; Sohmura, T.; Yamada, S.; Wakabayashi, K.; Nakamura, T.; Yatani, H. "Application of virtual reality force feedback haptic device for oral implant surgery." Clinical oral implants research, 17(6), 708-713, 2006.
- [12] Mcneely, W. A.; Puterbaugh, K. D.; Troy, J. J. "Six degree-of-freedom haptic rendering using voxel sampling." ACM SIGGRAPH 2005 Courses. p.42, 2005.

- [13] Morris, D.; Sewell, C.; Barbagli, F.; Salisbury, K.; Blevins, N. H.; Girod, S. "Visuohaptic Simulation of Bone Surgery for Training and Evaluation." IEEE Computer Graphics and Applications, v. 26, n. 6, p. 48-57, 2006.
- [14] Petersik, A.; Pflesser, B.; Tiede, U.; Hoehne, K. H.; Leuwer, R. "Haptic volume interaction with anatomic models at sub-voxel resolution." 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002. HAPTICS 2002. Proceedings. p.66-72, 2002. IEEE.
- [15] Rhiemora, P.; Gajananan, K.; Haddawy, P.; Dailey, M. N.; Suebnukarn, S. "Augmented reality haptics system for dental surgical skills training." Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology, p.97-98, 2010.
- [16] Ruspini D.;Kolarov K.;Khatib O."The haptic display of complex graphical environments." In Proc. of ACM Siggraph 97, Los Angeles, CA, pp. 345-352, 1997.
- [17] Salisbury, K.; Tarr. C. "Haptic rendering of surfaces defined by implicit functions." ASME Dynamic Systems and Control Division. Vol. 61. 1997.
- [18] Tsai, Ming-Dar; Ming-Shium Hsieh. "Accurate visual and haptic burring surgery simulation based on a volumetric model." Journal of X-ray Science and Technology 18.1 (2010): 69-85.
- [19] Vlasov, R.; Karl-Ingo F.; Franz-Erich W."Haptic Rendering of Volume Data with Collision Determination Guarantee Using Ray Casting and Implicit Surface Representation." Cyberworlds (CW), 2012 International Conference on. IEEE, 2012.
- [20] Wang, Changling Charlie; Charlie CL Wang. "Toward stable and realistic haptic interaction for tooth preparation simulation." Journal of Computing and Information Science in Engineering 10.2 (2010): 9.
- [21] Wang, Dangxiao; Yuru Zhang. "Effect of haptic device's position resolution on stability." Proceedings of EuroHaptics. 2004.
- [22] Wu, J.; Yu, G.; Wang, D.; Zhang, Y.; Wang, C. "Voxel-based interactive haptic simulation of dental drilling." International Design Engineering Technical Conferences & Computer and Information in Engineering Conference. ASME Press, California. 2009.
- [23] Zilles, C. B.; Salisbury, J. K. "A constraint-based god-object method for haptic display." Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on. v. 3, p.146-151, 1995.

Robust and Scalable Navmesh Generation with multiple levels and stairs support

Guillaume Saupin
CEA,LIST,Laboratoire de
Simulation Interactive
18 route du panorama
92260 Fontenay aux
Roses, France
guillaume.saupin@cea.fr

Olivier Roussel
CNRS,LAAS, Univ. de
Toulouse, UPS
7 avenue du colonel
Roche
F-31400 Toulouse,
France
olivier.roussel@laas.fr

Jeremie Le Garrec
CEA,LIST,Laboratoire de
Simulation Interactive
18 route du panorama
92260 Fontenay aux
Roses, France
jeremie.le-garrec@cea.fr

ABSTRACT

Automatically planning motion for robots or humans in a virtual environment is a complex task. The navigation cannot be done directly on the geometric scene. An internal representation of the environment is necessary. However, virtual environments complexity is growing at an important rate, and it is not unusual to work with millions of triangles and area as large as some square kilometers. In these cases, grid based methods require too much memory, and are too slow for A* algorithms. Polygons based methods are more memory and computation efficient, but they are difficult to generate and require complex preprocessing of the input mesh to ensure nice topological properties. In this paper, we propose an hybrid method, which uses 3D voxelization to generate a clean polygon mesh simple enough to perform fast A* search. Using this approach, we robustly generate *Navigation Meshes* for large environments with fine details, described by millions of triangles, without any assumption on the quality of the input mesh. The input mesh can contains holes, degenerated or intersecting triangles.

Keywords

navigation mesh generation, mesh simplification, path planning

1 INTRODUCTION

1.1 Motivations

Many tasks in computer animation require the computation of free paths. Computing them must be done in a very short time, as the requirement on the frame rate of this kind of application is very high.

To achieve such performances, path planning algorithms use internal representations detailed enough to model the complex virtual worlds, and simple enough to allow fast path request.

Computing these internal representations from the original meshes is a difficult task, virtual environments frequently being detailed, complex, large, and of various quality.

In this paper, we propose a method to efficiently and robustly compute polygon based maps, usually referred as *Navigation Meshes* from any triangles mesh.

1.2 Related works

As stated in apper [1], an important work has been done in the field of free path computation.

Lamarche [10] distinguish three approaches to path planning:

- Graph based methods, commonly referred as *roadmap* methods. The navigable environment is represented by a set of connected paths. They do not represent the entire environment and can lead to suboptimal paths. See papers [9] and [14].
- Cell decomposition methods, where the navigable environment is represented as a set of connected areas. The resulting map is referred as a *Navigation Mesh*. They were introduced by Snook [16].
- Potential fields methods, where the navigation is done according to the gradient of a potential field.

Paper [10] reviews of all these methods.

We choose to use *Navigation Meshes*, a good choice for applications requiring collision detection on environment boundaries, as well as path planning with arbitrary clearance.

1.2.1 Navigation Meshes

Path planning maps are usually based on two kinds of internal representations:

- Grid based discretization: the virtual environment is discretized using voxels.
- Polygon based discretization: the navigable areas are discretized using polygons.

In both cases, the discretization is associated with topological information regarding the connections between adjacent voxels or polygons.

Most of the proposed methods for path planning on *Navigation Meshes* are based on the seminal paper [3] introducing the A^* algorithm. This method has been applied with success on both polygonal [10] and voxels grids [1].

In the case of grid based method, generating the discretization and the topological information is a straightforward process. Some papers [2, 11, 12], propose implementation on GPU. However, for large environments with fine details, these method requires large amounts of memory, and became inefficient for path planning. This approach was introduced by Bandi in [1].

Polygons based methods are much more optimal internal representation. They require less memory and allow faster path finding. They can even be used for dynamic environments [17].

Many path requests can be done on such representations. Kallmann [7] explores the use of triangulation as *Navigation Meshes*. In paper [8] a method is proposed to compute shortest pathes with arbitrary clearance.

However, creating *Navigation Meshes* by hand is a tedious and error prone process, and automatically generate them from any input mesh is a challenging task.

Lamarche [10] has proposed a method using prismatic spatial subdivision, but it implies that all triangles intersections are materialized by edges. This assumption on the quality of the input mesh requires a potentially time consuming cleaning step.

Mononen [13] has published an open source implementation of *Navigation Mesh* generation similar to our method. He uses voxelization to extract walkable areas, and triangulate them to create the *Navigation Mesh*. However, this method is not scalable, and doesn't handle staircases or very large and detailed environment. He also doesn't guarantee that all the *Navigation Meshes* vertices lies on boundaries, which prevents the use of path planning algorithms with arbitrary clearance.

Finally, Oliva *et al.* [15] propose a method to automatically generate suboptimal *Navigation Meshes*, but they need clean polygons as an input.

We present a method that can robustly generate such polygonal maps from any input mesh, with low memory requirement, and in a matter of seconds. This method also handles ramps and staircases.

1.3 Contributions

In this paper we introduce new algorithms dedicated to produce a quality *Navigation Mesh* using a tiling process.

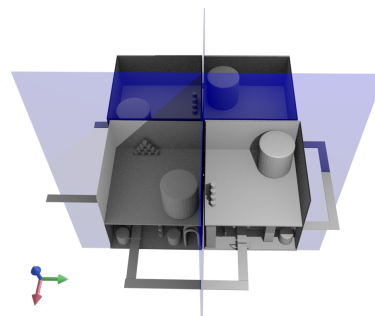


Figure 1: The input mesh has been split in four tiles.

In particular, we address the difficult problem of merging the regions segmented independently on each tile without introducing intermediate vertices. The proposed algorithms perform this operation robustly, while using a tiling process allows to perform a significant part of the computations in parallel as well as allowing to handle large 3D environments.

1.4 Method overview

Our method does not impose any requirement on the input triangles, and can be applied without any preprocessing. It ensures the following properties:

1. No assumption is made on the quality of the input mesh.
2. All the vertices of the *Navigation Mesh* are on the boundary. There is no intermediate vertices, introduced by the generation. This is a strong requirement on the quality of the *Navigation Mesh*, allowing to use complex navigation algorithms.
3. It handles large input, with area extending on square kilometers, and containing small details. Scenes can contain tenth of millions of triangles.
4. It does not required too much memory.
5. It is parallelized to ensure good performances.

To guarantee the first requirement, we choose to transform the input mesh in a voxel grid. This transformation can be processed on any input mesh. However, depending on the geometrical scene dimension, and the voxels size, it can use a large amount of memory.

To satisfy points 3 and 4, *i.e.* to support large input and consume a reasonable amount of memory we propose to handle the voxelization tile by tile. See Figure 1. This tile by tile processing also provides good performances on multi core architectures as its parallelisation is straightforward.

However, it introduces an important pitfall : point 2 states that we don't want any vertices to be artificially introduced in the *Navigation Mesh* by the method. All the vertices of the resulting mesh must belong either to the boundaries or to the obstacles. But the tiling process can add artificial vertices on the tiles boundaries.

We propose to carefully merge the tiles to avoid this problem.

This method boils down to six steps :

1. Filtering of the non navigable triangles of the input mesh
2. Voxelization of the filtered triangles
3. Tile based segmentation
4. Tile based contours extraction
5. Tiles merging region by region
6. Navigation graph building

The two first steps are detailed in section 2. The segmentation is described in section 3, while contour extraction is explained in section 4.1 and 4.2. Section 4.3 presents tile merging and section 5 graph building. Finally, results are discussed in Section 6.

2 VOXELIZATION

Most applications generating *Navigation Mesh* use as an input a geometry described by triangles. The quality of this triangulation depends on various factors: the geometric modeler, the designer, the triangulation process, the storage format, ... As a result, there is no guarantee on the topological properties of the input mesh.

In paper [10], a mesh cleaner is used to ensure these properties, but that implies the use of external libraries, and can be computationally intensive. And for very complex scenes, this cleaning can be intractable.

For these reasons, we design our navigation mesh generation method such that the input mesh doesn't have to satisfy any assumption. We don't perform any preprocessing on the mesh, and just need a list of triangles.

2.1 Tiling process

Instead, as Bandi [1] has proposed we voxelize the input mesh. However to ensure large input meshes processing, we work tile by tile, as shown in Figure 1. This limits the memory requirement, allows parallelization and authorizes generation of *Navigation Meshes* spreading on square kilometres with centimetric precision.

The tile size in $voxels^2$ can be set arbitrary, depending on the memory available on the computer and the algorithm implementation. In our applications, we usually choose a square tile of size between 256×256 and 1024×1024 $voxels^2$.

Note that splitting the input mesh in tiles creates artificial tiles boundaries that add artificial vertices which do not lie on real boundaries or obstacles. See Figure 6. This prevents the use of algorithms for planning with arbitrary clearance described in paper [8], as these algorithms require that the vertices of the triangulation belong to the obstacles.

Section 4.3 explains how to efficiently and robustly merge the tiles to remove these artificial points.

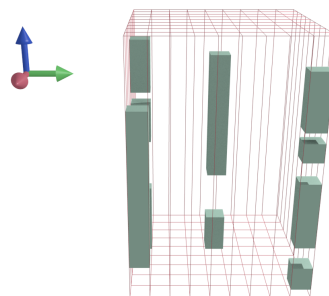


Figure 2: The vertical spans stored in a 2D grid.

2.2 Tiled Voxelization

To further improve the memory use, we don't use a 3D grid, but 2.5D grid, as in paper [13]. *I.e.* we use a 2D uniform grid in the X-Y plan, and each cell contains a list of vertical spans. See Figure 2. Each tile has its own 2.5D grid, whose each cell contains a list of spans.

Span have the following attributes:

- $z_{min} z_{max}$: min and max discrete altitude of the voxels span.
- l : a label identifying the associated region. Initially unset.

Note that z_{max} , the voxels span upper boundary might be a potential floor, whereas z_{min} might be a potential ceiling. The label l is originally undefined and is set in section 3, during the segmentation phase.

2.3 Filtering

2.3.1 Slope filtering

Before applying the tile based voxelization, it is interesting to filter the input to remove triangles that aren't navigable. For instance a man in a wheel chair cannot roll on a plan whose slope exceed 25%.

2.3.2 Tagging non navigable vertical spans

It is also important to tag non navigable vertical spans. For instance, the first $span_1$ of two consecutive spans with the same x, y coordinates must be tagged as non navigable if the distance $\Delta z = z_{min,span_2} - z_{max,span_1}$ is less than a robot height.

3 SEGMENTATION

Once the 2.5D voxelization has been performed, we segment the vertical spans in connected regions. The vertical spans coordinates $x-y$ are integers as well as their minimal and maximal $z_{min,max}$. This is important as it eases the segmentation process since we don't have to bother with additional epsilon parameters.

This segmentation is done on a per tile basis, and we choose to perform it slice by slice. This could introduce discontinuities in the regions, depending on the height of the vertical span, but these discontinuities can be easily removed as shown in section 5.2.

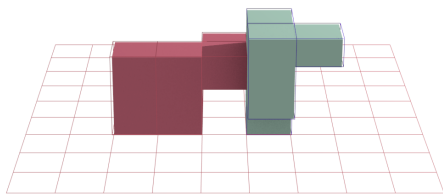


Figure 3: The green vertical spans are connected. They are neighbours and all have the same z_{max} . Some of the red vertical spans have green vertical spans as neighbours, but they don't have the same z_{max} , so they are not connected.

3.1 Flood fill

We use a very simple region growing algorithm to perform the segmentation. Mononen [13] uses the watershed based method proposed in paper [4], but we found this method to introduce over-segmentation in this context.

3.1.1 Connected vertical spans

Before proceeding to the segmentation itself, we must define the criteria satisfied by two vertical spans to be considered as connected:

1. They must have the same discrete z_{max} .
2. They must be neighbours, *i.e.* a connected vertical span must be one of the eight immediate neighbours of a vertical span.

These properties are summarized in Figure 3.

3.1.2 Segmentation algorithm

Knowing how to decide whether two vertical spans belong to the same zone, we can start the segmentation. The input of our algorithm is the 2.5D grid of vertical spans for the current tile. We use a flood fill. The basic idea of region growing algorithm is to start with a vertical spans, and to check whether his neighbours are connected. If this is the case, they belong to the same region and get the same label. The region growing is then continued using these connected neighbours.

We describe this simple algorithm in Algorithm 1 and 2.

Once the algorithm terminates, the list Z_s has been filled with lists of spans that are interconnected according to our criteria, and we are ready for extracting the contours of these regions.

Algorithm 1 Region growing based segmentation

Require: S filled with all the vertical spans of the current tile

```

1: while stack  $S$  is not empty do
2:   Pop vertical span  $v$  from the stack  $S$ 
3:   Set a new label  $l_{new}$  to  $v$ 
4:   Add an empty new list  $L_v$  of regions in  $Z_s$ .
5:   Add  $v$  to this new list  $L_v$ .
6:   Push  $v$  to  $Z_c$ 
7:   while  $Z_c$  is not empty do
8:     Pop vertical span  $v_i$  from  $Z_c$ 
9:     call handleNeighbours( $v_i, Z_c, L_v$ )
10:  end while
11: end while

```

Algorithm 2 Handle vertical span neighbours depending on their connexion

```

1: handleNeighbours( $v_i, Z_c, L_v$ )
2: for each neighbour cells  $c_n$  of  $v_i$  of the 2.5D grid do
3:   for each vertical span  $v_n$  of  $c_n$  do
4:     if  $v_n$  and  $v_i$  are connected then
5:       Set  $v_n$  label as  $l_{new}$ 
6:       Add  $v_n$  to list  $L_v$ .
7:       Push  $v_n$  into stack  $Z_c$ 
8:     end if
9:   end for
10: end for

```

4 MERGING TILE REGION

4.1 Corner Points Extraction

4.1.1 Identifying contour edges

The Figure 4 represents a typical region, with complex boundaries, holes, and connected holes. Contours connect the vertices stressed by the small spheres. These vertices can be easily identified by looking at the vertical spans connectivity. Let's call these vertices *corner points*. When looking at this region, two observations can be made:

1. If you browse contours rows by rows and columns by columns, it appears that there is an alternation in the connection between the *corner points*. Hence the first corner point of a row always connect to the second one ; the second one is never connected to the third ; the third always connected to the fourth ; ... This is in fact an application of the Jordan Theorem.
2. However, there is an exception to this rule: vertices shared by two holes. This is the case for the two red spheres. Let's call these vertices *slash* and *antislash* points depending on the holes position. See Figure 4.

4.1.2 Slash and Antislash Points

The third observation can be easily circumvented by identifying the *slash* and *antislash* and by duplicated them. Once duplicated, the second observation is always true. See Figure 7.

4

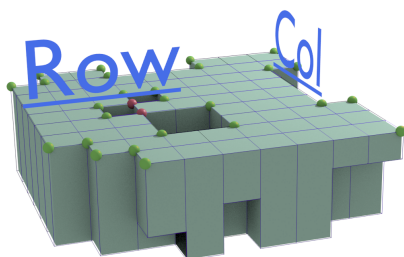


Figure 4: *Corner points* extraction. The upper red sphere is a *slash* point, as its surrounding holes form a *slash* diagonal. The lower red sphere is an *antislash* point, as its surrounding holes form an *antislash* diagonal.

4.1.3 Corners Points Extraction

Corner points are defined by their connectivity. As shown in Figure 5, the floor of each vertical span has 4 vertices, and these vertices are *corner points* if they are not surrounded by 4 vertical spans belonging to the same region. We define the neighbour configuration of a vertical span as the integer n_c on 8 bits encoding its neighbours.

For instance, a vertical span with no neighbours vertical spans has $n_c = 0$. A vertical span surrounded by vertical spans has $n_c = 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 = 255$. And a vertical span neighbours of vertical span 8 and 16 has $n_c = 8 + 16$.

Knowing this configuration, it is straightforward to identify the *corner points* using the following mask:

1. point 1 is corner if : $n_c \& 11 \notin \{11, 8, 2\}$
2. point 2 is corner if : $n_c \& 22 \notin \{22, 2, 16\}$
3. point 4 is corner if : $n_c \& 104 \notin \{104, 8, 64\}$
4. point 8 is corner if : $n_c \& 208 \notin \{208, 16, 64\}$

Slash and *antislash* points can also be detected similarly. These kind of points are *corner points*, and must pass the tests above as well as:

1. point 1 is a *antislash* point if : $n_c \& 11 = 1$
2. point 2 is *slash* point if : $n_c \& 22 = 4$
3. point 4 is *slash* point if : $n_c \& 104 = 32$
4. point 8 is *antislash* point if : $n_c \& 208 = 128$

Hence, to identify corners points, for each region of a tile, we iterate through all the spans, compute their neighbour configuration, apply the tests above, and store them region by region in L_c .

4.1.4 Handling vertices on the tile boundaries

The input mesh being processed tiles by tiles, some navigable regions lying on more than one tile are artificially cut into multiple regions. Regions merging is then required, and is described in section 4.3.

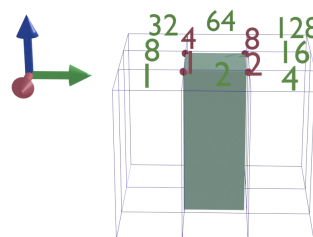


Figure 5: The 4 vertices of the vertical span floor are labelled 1,2,4 and 8, whereas the neighbour vertical spans are labelled 1, 2, 4, 8, 16, 32 and 128. Hence, the neighbour configuration of a vertical span can be encoded on 8 bits *i.e.* a unsigned char.

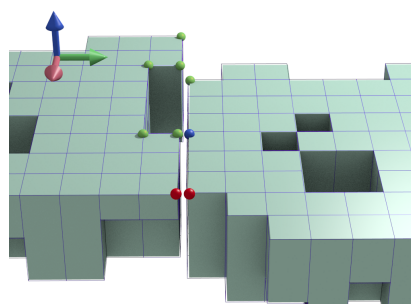


Figure 6: Handling tiles boundaries. Green and red spheres indicate corners point detected using the method above. Red ones must be removed as they are artificially added by tiling. Blue ones are added by taking into account neighbour tiles border vertical spans to ensure continuity.

However, to merge regions, we need to ensure some continuity on adjacent tiles boundaries. It is also important to distinguish vertices that are really *corner points* from vertices that are artificially labelled *corner points* due to the tiling process. See Figure 6.

Borders *corner points* can be classified in three categories :

1. Real *corner points*. These are the green spheres of Figure 6.
2. *corner points* if adjacent tiles are taken into account. These are the blue spheres of Figure 6.
3. False *corner points* that wouldn't exist without the tiling. These are the red spheres of Figure 6.

There is nothing special to do with the first class of points.

For the second and third classes of points, we must apply the method above on tile extended by a band of one vertical span on each of its four boundaries. Points of the second class will be detected by this operation, while points of the third class will be removed. It is then straightforward to identify these points by comparison with the *corner points* of the unextended tile.

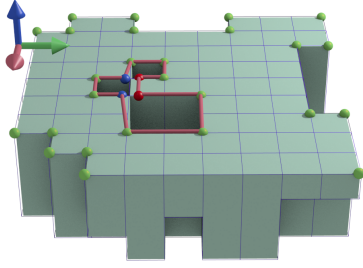


Figure 7: Contour extraction for duplicated *corner points*. Blue point is the first duplicate, while red one is the second. Vertical connection follow the rules of Table 1.

4.2 Contour Extraction

Once we have identified the *corner points* of the region extracted during the segmentation, we proceed to the extraction of the contour. This extraction is done region by region. Remember that all the vertices in a region have the same z coordinate.

Our algorithm use two main structures:

- A list *rows* of corner vertices, stored rows by rows
- A list *cols* of corner vertices, stored columns by columns

The idea behind this algorithm is to exploit the fact that we are using discrete points in conjunction with the *even-odd rule* given by the Jordan theorem. We process the vertices row by row, and col by col, and add them in the *rows* and *cols* structures. Special *slash* and *antislash* points are duplicated. See Algorithm 3.

Algorithm 3 Fill *rows* and *cols* lists.

Require: L_c : the list of *corner points* stored region by region

```

1: for each region  $r$  do
2:   for each corner point  $s(x, y)$  in  $r$  do
3:     if  $s(x, y)$  is a slash or antislash point then
4:       Add  $s$  twice to  $rows[y]$ .
5:       Add  $s$  twice to  $cols[x]$ 
6:     else
7:       Add  $s$  to  $rows[y]$ 
8:       Add  $s$  to  $cols[x]$ 
9:     end if
10:  end for
11: end for

```

first \ second	<i>normal</i>	<i>slash</i>	<i>antislash</i>
<i>normal</i>	$p \rightarrow p$	$p \rightarrow p_2$	$p \rightarrow p_1$
<i>slash</i>	$p_1 \rightarrow p$		$p_1 \rightarrow p_1$
<i>antislash</i>	$p_2 \rightarrow p$	$p_2 \rightarrow p_2$	

Table 1: Vertical connection between corner point depending on their type : *normal*, *slash*, *antislash*. p_1 and p_2 stands respectively for the first and second duplicate of *slash* or *antislash* points.

Then, we connect vertices by pair, horizontally, and vertically, using the *even-odd rule*, with special attention

for duplicated points. See Algorithm 4 and Table 1. As a result we get \mathcal{N}_h and \mathcal{N}_v that give respectively the horizontal and vertical neighbour of each vertices $v \in L_c$.

Algorithm 4 Build contour graph

Require: *rows* and *cols*

```

1: for each row of rows do
2:   Sort corner points of row in increasing  $x$  order.
3: end for
4: for each col of cols do
5:   Sort corner points of col in increasing  $y$  order.
6: end for
7: for each row of rows do
8:   for each  $n - 1$  corner points  $s_i$  in row do
9:     if  $i$  is even then
10:      Set  $\mathcal{N}_h(s_i) = s_{i+1}$ 
11:      Set  $\mathcal{N}_h(s_{i+1}) = s_i$ 
12:    end if
13:  end for
14: end for
15: for each col of cols do
16:   for each  $n - 1$  corner points  $s_i$  in col do
17:     if  $i$  is even then
18:       if  $s_i$  and  $s_{i+1}$  are not a slash neither a antislash points then
19:         Set  $\mathcal{N}_v(s_i) = s_{i+1}$ 
20:         Set  $\mathcal{N}_v(s_{i+1}) = s_i$ 
21:       else
22:         set  $s_i$  neighbour according to Table 1
23:       end if
24:     end if
25:   end for
26: end for

```

Finally, we connect all the vertices together to get all the contours of the region, including holes contour, and we store them in *conts*. See Algorithm 5.

Algorithm 5 Connect all contours vertices.

Require: L_c , \mathcal{N}_v and \mathcal{N}_h

Ensure: The list of contours *conts*

```

1: while all vertices in  $L_c$  have not been connected do
2:   Pick a non connected vertex  $s$  in  $L_c$ 
3:   Create a new empty contour cont, i.e. an empty list of vertices
4:   Add  $s$  into cont.
5:   Let  $s_c$  be the vertical neighbour of  $s$ 
6:   while  $s_c \neq s$  do
7:     Add  $s_c$  into cont
8:     if  $\mathcal{N}_h(s_c)$  already belongs to a contour then
9:       Set  $s_c = \mathcal{N}_v(s_c)$ 
10:    else
11:      Set  $s_c = \mathcal{N}_h(s_c)$ 
12:    end if
13:  end while
14:   Add cont to conts
15: end while

```

4.3 Regions merging

The algorithms used to extract the contours are performed tiles by tiles. This creates two difficulties :

- It artificially splits regions : as shown in Figure 6, if a navigable region of the input mesh lies on two or more tiles, it is cut in two or more unconnected regions.
- It creates artificial points. They augment the complexity of the resulting navigation mesh and preclude the possibility of using path planning with arbitrary clearance.

These difficulties can be circumvented by performing inter-tiles regions merging.

4.3.1 Vertices based merging

To reduce the complexity of the merging by one order of magnitude, we don't merge region on a vertical span basis, but on a border vertices basis. We proceed using an incremental approach. We process the regions of each tile sequentially.

Algorithm 6 Handle vertical spans neighbours depending on their connexion

Require: That each tile regions and contours have been computed

```

1: for each tile  $t$  do
2:   for each region  $r$  of  $t$  do
3:     Get the contour  $c$  of  $r$ 
4:     Get the vertices  $L_{v_3}$  of third class in the contour  $c$ 
5:     if some regions  $L_{sr}$  of  $L_r$  have some points in  $L_{v_3}$  then
6:       Merge the regions in  $L_{sr}$  in a unique region  $r_u$ 
7:       Add the region  $r_u$  to  $L_r$ 
8:     else
9:       Add the region  $r$  to  $L_r$ 
10:    end if
11:  end for
12: end for
13: for each region  $r$  of  $L_r$  do
14:   Remove the useless vertices of the third class
15: end for

```

First, we store the vertices artificially added by the tiling, those of the third class, in a dedicated array L_{v_3} . These vertices will be removed. If there is no such points, we create a new region, fill it with all the vertices and contour information, and add it to L_r .

Otherwise, we test these vertices, which by construction are shared with other regions, against the vertices of the region already stored in L_r . Again, if there is no such region, we create a new region, add all the vertices and contour information into, and add it to L_r . Otherwise, this gives us the list L_{sr} of regions whose contours share these vertices. We merge the regions in L_{rs} into a unique region r_u and add it to L_r . The regions listed in L_{rs} are removed from L_r .

Once all the regions of all the tiles have been processed this way, we iterate through the resulting regions, and remove the points of the third category.

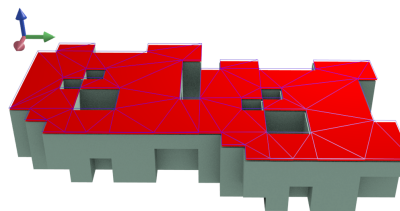


Figure 8: The triangulation for a unique region originally spreading on two different tiles.

See Algorithm 6 for a complete overview.

After this step, we have a list of regions defined by their vertices and their associated contours, and these regions are completely free from any artifact coming from the tiling process.

5 BUILDING THE NAVIGATION GRAPH

5.1 Triangulation

At this step, we have a list of unconnected and independent regions, defined by their contour, and which might contain holes. In order to display them, and apply our path planning algorithm, they must be triangulated.

This step is quite straightforward. We just have to identify the external contours, which surround the region, and the internal contours, which surround the potential holes.

5.1.1 Labelling contours

We sort the edges list $L_e(r)$ of each region r such that the first edge e_0 in $L_e(r)$ contains the bottom left vertex of the region contours.

Hence, using the Algorithm 7, we label the contours of the region so that the contour with label 0 is the external contour. All the others contours are internal contours surrounding holes.

5.1.2 Identifying holes edges

The Algorithm 7 also stores the bottom left vertex v_s for each contour. These vertices are used during the triangulation as seed to remove triangles inside holes. Figure 8 shows an example of region triangulation with holes.

5.1.3 Convex Partitioning

Finally, we use the simple convex partitioning algorithm presented in paper [6] to simplify our mesh by regrouping triangles into convex polygons. The method proposed in paper [15] can be used for a more optimal convex partitioning.

Algorithm 7 Label contours of region r **Require:** $L_e(r)$ filled with all the edges of the current region

```

1: Fill the stack  $S_e$  with the edges of  $L_e(r)$ 
2: Set the current contour label  $l_c = 0$ 
3: while  $S_e$  is not empty do
4:   Pop an edge  $e$  from the stack  $S_e$ 
5:   Let  $a_e, b_e$  be the vertices of  $e$ 
6:   Set  $a_e$  and  $b_e$  contour label to  $l_c$ 
7:   Set the bottom left vertices of current contour
    $\mathcal{L}_{over}(l_c) = a_e$ 
8:   Create an empty stack  $S_i$ 
9:   Find the edge  $e_a$  sharing  $a_e$  with  $e$ 
10:  if there is such a  $e_a$  edge then
11:    Remove  $e_a$  from  $S_e$ 
12:    Push  $e_a$  into  $S_i$ 
13:  end if
14:  Find the edge  $e_b$  sharing  $b_e$  with  $e$ 
15:  if there is such a  $e_b$  edge then
16:    Remove  $e_b$  from  $S_e$ 
17:    Push  $e_b$  into  $S_i$ 
18:  end if
19:  while  $S_i$  is not empty do
20:    Pop edge  $e_i$  from  $S_i$ 
21:    Let  $a_{e_i}, b_{e_i}$  be the vertices of  $e_i$ 
22:    Set  $a_{e_i}$  and  $b_{e_i}$  contour label to  $l_c$ 
23:    Update  $\mathcal{L}_{over}(l_c) = a_e$  according to vertices  $a_{e_i}, b_{e_i}$ 
    positions
24:    Find the edge  $e_{a_{e_i}}$  sharing  $a_{e_i}$  with  $e_i$ 
25:    if there is such a  $e_{a_{e_i}}$  edge then
26:      Remove  $e_{a_{e_i}}$  from  $S_e$ 
27:      Push  $e_{a_{e_i}}$  into  $S_i$ 
28:    end if
29:    Find the edge  $e_{b_{e_i}}$  sharing  $b_{e_i}$  with  $e_i$ 
30:    if there is such a  $e_{b_{e_i}}$  edge then
31:      Remove  $e_{b_{e_i}}$  from  $S_e$ 
32:      Push  $e_{b_{e_i}}$  into  $S_i$ 
33:    end if
34:  end while
35:  Increment current contour label  $l_c = l_c + 1$ 
36: end while

```

This algorithm is applied to all the regions detected in the previous step.

5.2 Connection Graph

After the partitioning, our input mesh has been transformed into a list of unconnected regions. To enable navigation between these regions, we must build a navigation graph linking the regions, to allow navigation on the whole navigation mesh

This navigation graph will be used for path planning using the A* algorithm [3].

5.2.1 Reconnect discontinuities due to discretization

Due to the voxelization process, some regions that were originally connected might have been artificially disconnected depending on the voxel height. These regions must be reconnected.

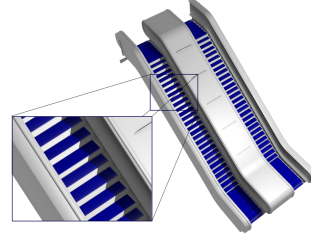


Figure 9: The blue polygons are the *Navigation Mesh* generated for this escalator.

Two regions that have been disconnected by the voxelization must have edges in common. The vertices of these edges have the same x, y discrete coordinates, and their discrete z coordinate must differ by only 1 voxel height.

Hence, to reconnect these regions, we simply add a link in the navigation graph that connects two polygons of two different regions which have each an edge satisfying the constraint above.

5.2.2 Stairs handling

Handling stairs is quite similar to disconnected region reconnection. See Figure 9. Two regions r_a and r_b are identified as connected stairs if there exists two edges $e_a \in r_a$ and $e_b \in r_b$ such that :

- $\Delta z < \max_{climb}$
- $\|v1_{x,y}^a - v1_{x,y}^b\| < r$ and $\|v2_{x,y}^a - v2_{x,y}^b\| < r$

Where $v1^a$ and $v2^a$ are the two vertices of e_a , $v1^b$ and $v2^b$ are the two vertices of e_b , and Δz is the difference between the z coordinates of e_a and e_b . Note that $v1$ and $v2$ have the same z coordinate when they belongs to the same region.

A link is added in the navigation graph for each two polygons whose one of the edges satisfy these requirements.

6 RESULTS

6.1 Settings used

Our method uses only three parameters :

- The voxel dimensions: c_x, c_y, c_z
- The maximum height of a step: \max_{climb}
- The tile size in voxels: n_{vx}

To evaluate the robustness of our method, we choose 5 different settings for these parameters. See Table 2.

The default set uses sensible parameters, that allow a good precision while keeping the number of voxels at a reasonable level.

The setting S2 use small voxels, with exotic dimensions.

Table 2: The settings used to test the robustness of our method

Set.	c_x	c_y	c_z	max_{climb}	n_{vx}
def.	0.05	0.05	0.02	0.5	256
S2	0.0356	0.0356	0.02	0.5	256
S3	0.0666	0.0666	0.5010	0.5	256
S4	0.666	0.666	0.587	0.5	256
S5	0.05	0.05	0.02	0.5	467

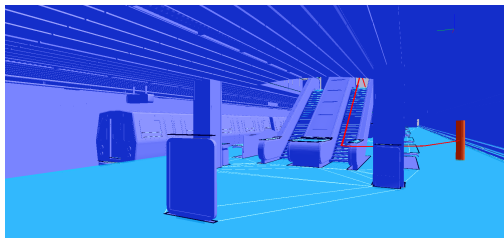


Figure 10: The subway station using the default settings. The navigation mesh is represented in light blue. An example of path found using the escalator is shown by the red line. The path planning query took 1.329ms.

Table 3: Results for the subway station

Set.	#polys	#voxels	#links	time (s)
def.	12 288	7021 * 804 * 415	18 088	14.122
S2	11 550	9861 * 1129 * 443	24 358	26.035
S3	2342	5271 * 603 * 17	4 334	5.314
S4	286	527 * 60 * 14	650	0.726
S5	11 188	8776 * 1005 * 553	23 916	19.79

The setting S3 use high voxels with a tight basis, also with exotic dimensions. Moreover, the voxels dimension c_z is bigger than the value max_{climb} . This means that it will be impossible to detect steps.

The setting S4 use big voxels, whose dimension c_z is bigger than the value max_{climb} . This setting should create coarser navigation mesh, with less polygons, but in a shorter time, as less vertical spans are created.

Finally, the setting S5 just changes the size of the tiles. This should just change the performances.

6.2 Navigation Mesh generation results

The 5 settings presented above have been tested on various meshes. We choose to present the results for 3 representative meshes. See Figure 10, 11 and 12.

A coherent navigation mesh have been generated for each pair settings/3D scene.

6.2.1 Subway station

This scene has 568 418 triangles. The results of the navigation meshes generation are summarized in Table 3.

The navigation meshes have been generated in a short time for all settings. The resulting navigation meshes are very simple, and contain a small number of polygons in comparison with the original 568 418 triangles.

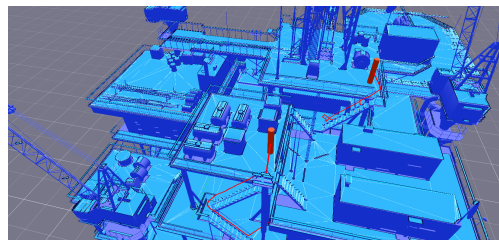


Figure 11: The platform using the default settings. This is a complex scene on multiple levels, with many details : pipe, stairs, cranes, ... An example of path found on multiple level using the stairs is shown by the red line. The path planning query took 4.836ms.

Table 4: Results for the platform.

Set.	#polys	#voxels	#links	time (s)
def.	21 331	800 * 1045 * 2404	41 864	9.727
S2	32 976	1124 * 1468 * 2572	65 488	12.586
S3	9 511	601 * 785 * 96	18 834	3.339
S4	537	60 * 78 * 82	972	0.097
S5	30 424	1000 * 1307 * 3206	59 616	15.812

The results are coherent with the parameters. Smaller voxels leads to more detailed navigation meshes, with more polygons. Using smaller voxels also increased the generation time.

As shown in figure 10, the escalator have been detected with the defaults parameters and the settings S2 and S5. The detection of stairs allows to use walking algorithms like the one described in paper [5], which was impossible in *Recast* [13].

6.2.2 Platform

The scene platform has 119 601 triangles. The results of the navigation meshes generation are summarized in Table 4.

This scene is a complex one, with multiples interconnected levels. There is a lot of details, with many pipes, stairs, cables, cranes, ... There are also intersecting triangles so the method proposed by Lamarche [10] would not work on this 3D scene without prior cleaning. Cleaning a complex scene like this one can be tedious and remove important details.

Moreover, in paper [10], the *House* scene has a complexity similar to this *Platform* scene, *i.e.* 120 160 triangles, and they create the navigation mesh in 904.32s. Our method is about 10 times faster.

The stairs have been detected with the defaults parameters and the settings S2 and S5.

6.2.3 Buildings

The scene building has 1 121 127 triangles. See Table 5 for all results.

This scene is a not very complex, but is very large. Its dimensions are about $500 * 500 * 16m^3$. This lead to

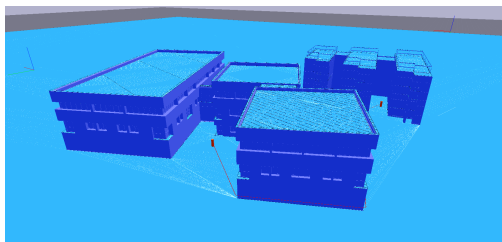


Figure 12: The buildings scenes using the default settings. It's a very large scene. The path planning query took 5.581ms.

Table 5: Results for the buildings.

Set.	#polys	#voxels	#links	time (s)
def.	117 488	8463 * 10977 * 804	235 964	229.854
S2	180 173	11886 * 15417 * 860	364 132	612.833
S3	54 139	6353 * 8241 * 32	107 122	63.863
S4	2 622	635 * 824 * 27	4 980	1.591
S5	164 577	10579 * 13721 * 1073	333 554	400.257

a huge number of voxels when using small voxels dimension. Using the default parameters, we manage to create the navigation mesh in less than 4 minutes. Pure voxels method like the one detailed in paper [1] can't handle scene large as this one with a resolution as low as $0.05*0.05*0.02 m^3$ as the number of voxels make it impossible to compute the path planning in a reasonable time. The memory requirement, when no tiling is used, would also prevent the use of the method proposed in paper [1].

Our method successfully creates the navigation meshes for each settings, and we were able to handle path planning request in less than 5ms.

7 CONCLUSION

In this paper we have detailed all the necessary steps to robustly generate *Navigation Meshes* from any triangles scene. Our method support very large scenes, with stairs, and uneven terrains. Absolutely no assumption is made on the quality of the input mesh. The resulting *Navigation Meshes* support path planning with arbitrary clearance. Moreover, the performances are excellent, as we are about 10 time faster than the method presented in paper [10].

We have applied this method with success in various context: serious gaming, security, virtual human walk, robot control, or crowd simulation. We plan to further improve our algorithm by supporting Multi Resolution Analysis, and by automatically setting the voxels size depending on the level of details in each tile. We will also explore hierarchical representation of uneven terrains in order to reduce navigation graph complexity.

8 REFERENCES

[1] Srikanth Bandi and Daniel Thalmann. Space discretization for efficient human navigation. *Computer Graphics Forum*, 17(3):195–206, 1998.

- [2] Elmar Eisemann and Xavier Décoret. Single-pass GPU Solid Voxelization and Applications. In *GI '08: Proceedings of Graphics Interface 2008*, volume 322, pages 73–80, Windsor, ONT, Canada, 2008. Canadian Information Processing Society.
- [3] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, july 1968.
- [4] Denis Haumont, Olivier Debeir, and François X. Sillion. Volumetric Cell-and-Portal Generation. In *Computer Graphics Forum*, volume 3-22 of *EUROGRAPHICS Conference Proceedings*, pages 303–312, Grenade, Espagne, 2003. Blackwell Publishers.
- [5] Andrei Herdt, Holger Diedam, Pierre-Brice Wieber, Dimitar Dimitrov, Katja Mombaur, and Moritz Diehl. Online Walking Motion Generation with Automatic Foot Step Placement. *Advanced Robotics -Utrecht-*, 24(5-6):719–737, 2010.
- [6] Stefan Hertel and Kurt Mehlhorn. Fast triangulation of simple polygons. In Marek Karpinski, editor, *Foundations of Computation Theory*, volume 158 of *Lecture Notes in Computer Science*, pages 207–218. Springer Berlin Heidelberg, 1983.
- [7] Marcelo Kallmann. Navigation Queries from Triangular Meshes Motion in Games. volume 6459 of *Lecture Notes in Computer Science*, chapter 22, pages 230–241. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2010.
- [8] Marcelo Kallmann. Shortest paths with arbitrary clearance from navigation meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 159–168, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.
- [9] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, aug 1996.
- [10] Fabrice Lamarche. Topoplan: a topological path planner for real time human navigation under floor and ceiling constraints. *Comput. Graph. Forum*, pages 649–658, 2009.
- [11] Duoduo Liao. Gpu-accelerated multi-valued solid voxelization by slice functions in real time. In *Proceedings of the 24th Spring Conference on Computer Graphics, SCCG '08*, pages 113–120, New York, NY, USA, 2010. ACM.
- [12] Ignacio Llamas. Real-time voxelization of triangle meshes on the gpu. In *ACM SIGGRAPH 2007 sketches*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.
- [13] Mikko Mononen. Recast navigation, May 2012.
- [14] D. Nieuwenhuisen, A. Kamphuis, and M. H. Overmars. High quality navigation in computer games. *Sci. Comput. Program.*, 67(1):91–104, June 2007.
- [15] Ramon Oliva and Nuria Pelechano. Automatic generation of suboptimal navmeshes. In *Proceedings of the 4th international conference on Motion in Games*, MIG'11, pages 328–339, Berlin, Heidelberg, 2011. Springer-Verlag.
- [16] Greg Snook. Simplified 3d movement and pathfinding using navigation meshes. *Game Programming Gems*, pages 288–304, 2000.
- [17] Wouter van Toll, Atlas F. Cook IV, and Roland Geraerts. A navigation mesh for dynamic environments. *Journal of Visualization and Computer Animation*, 23(6):535–546, 2012.

Single view single light multispectral object segmentation

Eduard Schick	Steffen Herbot	Arne Grumpe	Christian Wöhler
TU Dortmund University Image Analysis Group Otto-Hahn-Straße 4 44227, Dortmund, Germany	TU Dortmund University Image Analysis Group Otto-Hahn-Straße 4 44227, Dortmund, Germany	TU Dortmund University Image Analysis Group Otto-Hahn-Straße 4 44227, Dortmund, Germany	TU Dortmund University Image Analysis Group Otto-Hahn-Straße 4 44227, Dortmund, Germany
eduard.schick@tu-dortmund.de	steffen.herbot@tu-dortmund.de	arne.grumpe@tu-dortmund.de	christian.woehler@tu-dortmund.de

ABSTRACT

In this paper we present an approach for the acquisition and segmentation of spectral Bidirectional Reflectance Distribution Function (BRDF) measurements of real-world objects. The acquisition setup is a priori fully calibrated and provides pixel-synchronous image and depth data of the examined objects. Based on one single viewing and illumination geometry, we are able to determine spectrally distinct surface regions for objects with abruptly changing surface materials (painted surface patches) and for objects with gradually changing materials (partially oxidized iron). For clustering we apply the k -means algorithm and the mean-shift algorithm. The segmented clusters are used to adapt individual spectral BRDFs (Lambert, Phong, Cook-Torrance) to the obtained cluster data. Additionally, the elemental abundances of iron and rust on a metal surface are analyzed using spectral unmixing. The paper presents a detailed discussion of our method and provides critical insight into the obtained results.

Keywords

multispectral data, segmentation, k -means, mean-shift, BRDF, linear unmixing

1 INTRODUCTION

The amount of light perceived from a point on an arbitrary illuminated object surface depends on the type and direction of the incident light, the surface geometry, the scene geometry, and, especially, on the object material at any given point. Industrial applications show that it is desirable to develop methods for the determination of regions with common or distinct spectral reflectance behavior. The knowledge of local material properties can then be used for e.g. material identification or image based surface reconstruction [HB86, Woo80, HW12]. While successful material separation has been presented by different researchers (e.g. [Tom02, NRN03, LKG⁺03]), their data acquisition involved a varying light source position and a varying camera position, which requires an intricate measurement scenario involving a high effort for data acquisition.

In this paper, we present a method for the separation of material components based on their multispectral reflectance characteristics using a static object, a single light source position, and a fixed camera position. For industrial applications, it is desirable to use as few light and camera positions as possible, since this facilitates the acquisition process and reduces the required recording time. However, this comes at the cost of the measurements covering only a small range of possible angles of observance ($\vartheta_o \in [-90^\circ \dots +90^\circ] \subset \mathbb{R}$) and in-

cidence ($\vartheta_i \in [-90^\circ \dots +90^\circ] \subset \mathbb{R}$). While it is challenging to cluster and/or perform model fits using that data basis, we will show that it is possible if the underlying models are modified. We are thus able to obtain correct segmentations for almost the whole object, even though there is no wide range of illumination and viewing angles available due to the single view single light configuration.

We compare the application of different clustering approaches (k -means clustering and mean-shift clustering) for the examination of a surface with sharp material changes (color patches), and for a surface with smooth material changes (iron vs. iron oxide). For the segmentation, it is physically impossible to cluster the data based on specular reflections, since these only depend on the color of the incident light and are independent of the surface color as explained in detail by [Sha85]. The segmentation can thus be performed on the diffuse part only. We show that it is possible to improve the segmentation results considerably by an empirical modification of the modeled reflectance behavior.

In the subsequent stage we use each cluster to obtain the parameters of a model-based spectral BRDF¹, and we compare the performance of the Phong [Pho75] and Cook-Torrance [CT81] models. Additionally, we show

¹ Bidirectional Reflectance Distribution Function

that the clustered data can be used to determine the abundance distribution of – in our case – iron and iron oxide using linear unmixing without any adjustment of the underlying BRDF.

2 RELATED WORK

The problem of separating material from shape and illumination has been examined by some authors previously. However, most of them use some sort of varying light source position and/or varying viewpoint.

[Tom02] applies multispectral data to the problem of circuit board element segmentation. Initially, two images are recorded with different illumination directions, which are used for a very coarse initial segmentation and measurements that contain specular reflections are removed. The remaining pixels are clustered based on a set of empirically defined if-then-else classification rules, which emerge from a priori knowledge of the spectral reflectance properties of the circuit board elements. In summary, [Tom02] operate on the diffuse part only, require different illumination directions, and rely on a priori knowledge for the clustering process.

[NRN03] present “photometric invariants”, which can be used for various tasks, including material segmentation. In detail, they aim for a separation of material from lighting and shape based on a scene description with separable BRDFs. The availability of multispectral data (color images) under different illumination conditions then allows for solving for a geometric invariant by simply computing ratios of matrix determinants. The authors show successful application of their method to isolated homogeneous objects, homogeneous objects in complex environments, objects consisting of different materials, inhomogeneous objects, and objects with a specular surface. However, the invariants require changes in illumination, viewpoint, and/or object position between consecutive acquisitions.

[LKG⁺03] acquire geometric and photometric data using different light source positions and viewpoints for a set of “lumitexels”, which are assembled as a vector of geometric and photometric measurements. Afterwards, they fit the Lafortune BRDF model to all lumitexels to create two BRDF models along with a covariance matrix of the fitting parameters. Based on the error between lumitexels and those two BRDFs, they split the surface consecutively into two clusters. This procedure is repeated for the cluster with the greatest deviation of measured and fitted BRDF until a clear segmentation of all material components is achieved.

3 CONTRIBUTION

We present a novel method for the segmentation of multispectral data. In contrast to [LKG⁺03] and [NRN03], we apply a fixed object, light source, and camera position. Unlike [Tom02], we do not require

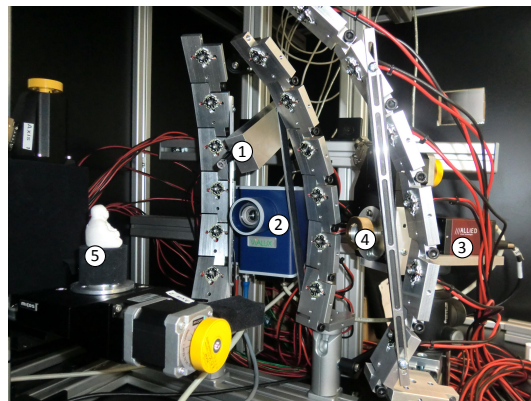


Figure 1: Experimental setup including light source (1), camera (3), 3D scanner (2+3) and interference bandpass filters (4) for acquisition of the object data (5).

any prior knowledge of the material reflectance. Fig. 1 shows an image of the experimental setup and Fig. 2 gives an overview of our algorithm. Initially, we record

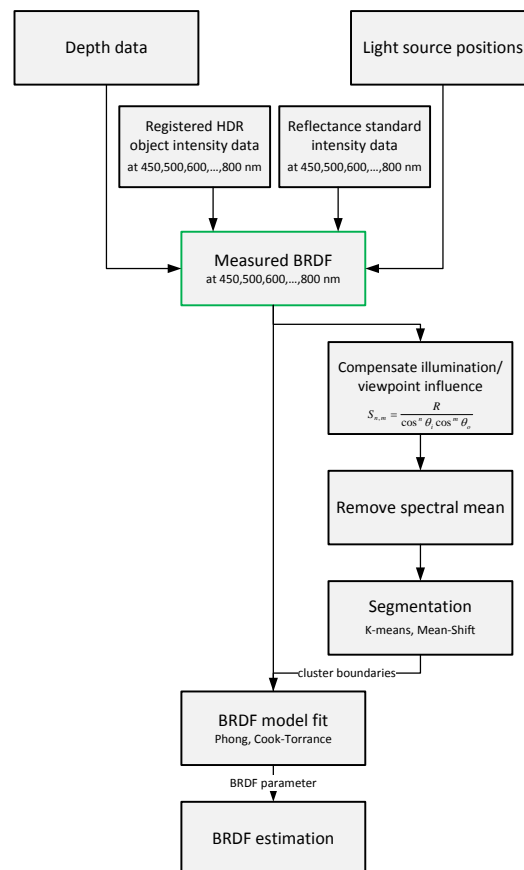


Figure 2: Methodical overview.

the depth data with a 3D laser pattern projector² and acquire image data for $N = 8$ distinct wavelengths

² ViALUX zSnapper Vario, 2048 px × 2048 px, lateral resolution $\approx 40 \mu\text{m}$

between 450 and 800 nm) using bandpass interference filters³. The light source positions and intensities are known a priori following the calibration procedure described by [LHW12]. Additionally, the image data of a white diffuse reflectance standard⁴ is recorded, which is later used to infer the BRDF from the observed intensity data (Section 4).

Afterwards, compensate the influence of the incident and viewing directions is compensated, the spectral mean is removed, and the k -means and/or mean-shift algorithms are employed for clustering. With the clustering boundaries available, we are able to obtain BRDF parameters for each cluster and each multispectral channel within these clusters. Algorithmic details for these stages are explained in the following section.

4 ALGORITHMIC DETAILS

4.1 Data preparation

In order to account for even slight camera shifts during bandpass filter replacements or by the optically active bandpass filters themselves, we conduct an image registration and subsequent affine transformation step to account for these distortions. The dynamic range of the images is increased by using high dynamic range imaging (HDR), which allows capturing bright specular regions and dark surface regions without reaching saturation or losing low-contrasted details in the camera noise.

Since the depth data are corrupted by a considerable amount of noise, we perform a model-based fit to the acquired data. The models are chosen to simply match the object shape, i.e. a cylindrical model for the “cup” dataset and a plate for the “triangle” dataset (see Fig. 3 and Fig. 10). This is achieved by a minimization of the mean squared distance between the measured data points and the model surface. A whole description of that process lies beyond the focus of this paper and is therefore omitted, a detailed description can be found e.g. in [LMM98, Ebe08]. Note, that the multispectral data segmentation algorithm below is not restricted to the application of these models. The models only account for the partially severe amount of noise present in our 3D scans. The model fitting step is unnecessary, if the scanned surface contains only low noise.

4.2 BRDF measurement

The main idea for BRDF measurement determination lies in acquiring multispectral object and reflectance standard data, which are then related to each other as

explained in the following. This derivation thus aims for an expression of the observed BRDF values f_r . The measured object intensity can be expressed as

$$I_{ob} = C \cdot \frac{I_0}{r_{l,ob}^2 r_{c,ob}^2} \cdot f_{r,ob} \cdot \cos \vartheta_{i,ob}, \quad (1)$$

where the distance from the light source to the object and the camera to the object are denoted $r_{l,ob}$ and $r_{c,ob}$, respectively. The light source intensity is I_0 , the BRDF is $f_{r,ob}$. Further influencing quantities like camera gain or interference filter attenuation are all contained within the constant C . $\vartheta_{i,ob}$ denotes the angle of light incidence on the object surface, i.e. the angle between the local direction \mathbf{l} of the incident light and the local normal vector \mathbf{n}_{ob} of the surface such that $\cos \vartheta_{i,ob} = \langle \mathbf{l}, \mathbf{n}_{ob} \rangle$. Analogously, the reflectance standard is described by

$$I_{re}(x_0, y_0) = C \cdot \frac{I_0}{r_{l,re}^2 r_{c,re}^2} \cdot f_{r,re} \cdot \cos[\vartheta_{i,re}(x_0, y_0)] \quad (2)$$

In contrast to Eq. (1), we can include more prior knowledge here. For the reflectance standard, we can specify an approximately diffuse reflectance behavior and an a priori known spectral albedo $\rho_{re} = 0.99$ specified by the manufacturer. We thus replace the general BRDF $f_{r,re}$ by a Lambertian reflectance term, i.e. $f_{r,re} = \frac{\rho_{re}}{\pi}$, yielding

$$I_{re}(x_0, y_0) = C \cdot \frac{I_0}{r_{l,re}^2 r_{c,re}^2} \cdot \frac{\rho_{re}}{\pi} \cdot \cos[\vartheta_{i,re}(x_0, y_0)]. \quad (3)$$

Dividing Eq. (1) by Eq. (3) then leads to

$$f_{r,ob} = \frac{I_{ob}}{I_{re}} \cdot \frac{\rho_{re}}{\pi} \frac{\cos \vartheta_{i,re}(x_0, y_0)}{\cos \vartheta_{i,ob}} \cdot \frac{r_{l,ob}^2 r_{c,ob}^2}{r_{l,re}^2 r_{c,re}^2}, \quad (4)$$

which is connected to the reflectance R_{ob} by

$$f_{r,ob} = R_{ob} \cdot \frac{1}{\cos(\vartheta_{i,ob})}. \quad (5)$$

It is apparent that the relation of measured object intensities with reflectance standard data provides an elegant way for BRDF measurement determination. The BRDF obtained from Eq. (4) can now be used to segment areas based on their spectral reflectance. Ideally, the separated areas correspond to areas with the same material characteristics. Practically, a specular reflection within the surface leads to incorrect segmentation results. Due to the lack of varying light source or viewpoint position, a precise estimation of these illumination effects can not be achieved. Instead, we modify the measured reflectance samples R_{ob} (or, respectively, measured BRDF f_r) such that

$$S_{n,m} = \frac{R_{ob}}{\cos^n \vartheta_i \cos^m \vartheta_o} = \frac{f_r}{\cos^{n+1} \vartheta_i \cos^m \vartheta_o}, \quad (6)$$

³ Thorlabs bandpass interference filters, center wave length CWL = [450, 500, 550, 600, 650, 700, 750, 800] nm, full width at half maximum FWHM = 10 nm

⁴ SphereOptics Zenith Polymer Diffuse Reflectance Standard SG3052

which has been found to alleviate that effect. The angles $\vartheta_i = \arccos(\langle \mathbf{l}, \mathbf{n} \rangle)$ and $\vartheta_o = \arccos(\langle \mathbf{v}, \mathbf{n} \rangle)$ denote the angles between incident light direction \mathbf{l} and observance direction \mathbf{v} with the local surface normal \mathbf{n} , respectively. Additionally, we apply a spectral mean subtraction to Eq. (6) prior to segmentation since this has been found to improve the segmentation results (see Section 5) because it stresses the spectral gradient in comparison to the spectral offset.

4.3 Clustering and BRDF fitting

For clustering similar BRDF measurements, we use the k -means algorithm [Mar09] and the mean-shift algorithm [CM02] in their classical form, i.e. without functional adaptations. For the mean-shift algorithm, we apply a disk-shaped kernel that evenly weights the involved measurements [Fin06]. Once the clusters are obtained, we estimate a Phong [Pho75] and a Cook-Torrance [CT81] BRDF by nonlinearly fitting the $u = [1 \dots U] \subset \mathbb{N}$ observed measurements $f_{r,u}^{\text{observed}}$ to the corresponding model based estimations $f_{r,u}^{\text{modeled}}$ such that

$$P^* = \underset{P}{\operatorname{argmin}} \sum_{u=1}^U (f_{r,u}^{\text{observed}} - f_{r,u}^{\text{modeled}})^2 \quad (7)$$

becomes minimized with respect to the model parameters P . Both BRDF models include a Lambertian term $\frac{k_d}{\pi}$ to describe (ideal) diffuse reflection. The applied physically plausible Phong BRDF is

$$f_r = \frac{k_d}{\pi} + k_s \frac{a+2}{2\pi} \underbrace{\langle \mathbf{r}, \mathbf{v} \rangle^a}_{\cos^a \vartheta_r}, \quad (8)$$

where a denotes an exponential coefficient for varying the angular extent of the specular component. The specular characteristics are modeled based on the angle ϑ_r between the direction of mirror reflection \mathbf{r} and observation \mathbf{v} .

The Cook-Torrance BRDF model consists the following terms:

$$f_r = k_d \frac{1}{\pi} + k_s \frac{F(n)}{\pi} \frac{D(m) \cdot G}{\cos \vartheta_i \cos \vartheta_o}. \quad (9)$$

The amount of reflected light depends of the three components F , D , and G , which denote the Fresnel reflection coefficient, the distribution function of the directions of the microfacets, and the geometrical attenuation factor, respectively. F improves the model exactness for grazing incidence angles, D defines the influence of surface roughness on lobe width, and G accounts for self shadowing and occlusion. The parameters of the Cook-Torrance BRDF are the weights k_d and k_s , the index of refraction n and the surface roughness m . Details can be obtained from [CT81].

4.4 Spectral unmixing

Spectral unmixing denotes the decomposition of the spectrum of a compound material into the spectra of the known pure materials out of which it consists by estimating the corresponding relative frequencies [KM02]. To estimate a distribution of a compound of two materials (“endmembers”) – in our case: iron and iron oxide (rust), see Fig. 10(a) – we use a linear unmixing approach [KM02] such that the pixel-wise spectral BRDF $\mathbf{f}_r = [f_{r,1}, \dots, f_{r,N}]^T \in \mathbb{R}^{N \times 1}$ sampled at N distinct wavelengths is supposed to be composed of two components $\mathbf{x}_{\text{iron}} \in \mathbb{R}^{N \times 1}$ and $\mathbf{x}_{\text{rust}} \in \mathbb{R}^{N \times 1}$ according to

$$\mathbf{x}_{\text{iron}} + \mathbf{x}_{\text{rust}} = \mathbf{BRDF} = \mathbf{f}_r. \quad (10)$$

These \mathbf{x} , in turn, are weighted variants of the spectral BRDFs of the pure elements (\mathbf{r}_{iron} and \mathbf{r}_{rust}):

$$\mathbf{x}_{\text{iron}} = a_{\text{iron}} \cdot \mathbf{r}_{\text{iron}} \quad \mathbf{x}_{\text{rust}} = a_{\text{rust}} \cdot \mathbf{r}_{\text{rust}}. \quad (11)$$

The vectors \mathbf{r}_{iron} and \mathbf{r}_{rust} represent reference spectra of iron and rust, which have been determined by averaging $K \in \mathbb{N}$ manually chosen BRDF samples $r_{\text{iron},k}$ and $r_{\text{rust},k}$ with $k = 1 \dots K$ from the image:

$$\mathbf{r}_{\text{iron}} = \frac{1}{K} \sum_{k=1}^K \mathbf{r}_{\text{iron},k}, \quad \mathbf{r}_{\text{rust}} = \frac{1}{K} \sum_{k=1}^K \mathbf{r}_{\text{rust},k}. \quad (12)$$

This approach corresponds to the concept of image-based endmember selection [KM02]. To estimate the actual abundance distribution $a_{\text{iron}} \in [0 \dots 1] \subset \mathbb{R}$ and $a_{\text{rust}} \in [0 \dots 1] \subset \mathbb{R}$, we apply the following least-square transformation of Eq. (11):

$$a_{\text{iron}} = (\mathbf{r}_{\text{iron}}^T \mathbf{r}_{\text{iron}})^{-1} \mathbf{r}_{\text{iron}}^T \mathbf{x}_{\text{iron}} \quad (13)$$

$$a_{\text{rust}} = (\mathbf{r}_{\text{rust}}^T \mathbf{r}_{\text{rust}})^{-1} \mathbf{r}_{\text{rust}}^T \mathbf{x}_{\text{rust}} \quad (14)$$

For this to be physical meaningful, the following two constraints have to be satisfied:

$$a_{\text{iron}}(i) > 0, \quad a_{\text{rust}}(i) > 0 \quad (15)$$

$$a_{\text{iron}}(i) + a_{\text{rust}}(i) = 1 \quad (16)$$

These constraints are added to Eq. (13) and Eq. (14) [KM02], which are then solved accordingly. Note that this derivation can be applied to the separation of two arbitrary materials with known or measured pure material BRDFs $\mathbf{r}_{\text{Material 1}}$ and $\mathbf{r}_{\text{Material 2}}$. In that case, the BRDF vectors \mathbf{r}_{iron} and \mathbf{r}_{rust} are replaced by the new BRDF vectors in the derivation above. The linear unmixing approach is also applicable to $N \in \mathbb{N} > 2$ material components.

5 EXPERIMENTS AND RESULTS

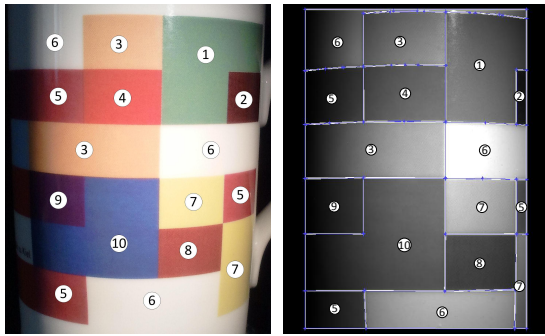
The following sections provide a detailed discussion of the “cup” dataset, which contains sharp boundaries (see Section 5.1), and of the “triangle” dataset, which exhibits a gradual transition between iron and iron oxide (see Section 5.2).

5.1 Object with sharp region boundaries

The first part of this work deals with the segmentation of a dataset with abruptly changing regions on the surface of a cup as shown in Fig. 3(a). To compare segmentation results quantitatively, we manually created a reference cluster map, which is shown in Fig. 3(b). With that being available as a ground truth, the detection rate D is defined to be

$$D = \frac{tp}{all} \cdot 100\%, \quad (17)$$

i.e. the percentage of correctly classified pixels (tp) within the whole set of classifiable pixels in the image (all).

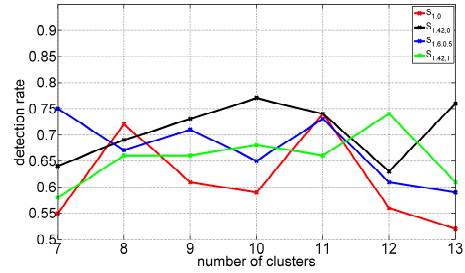


(a) Color regions of the examined cup (b) Manually defined reference clusters

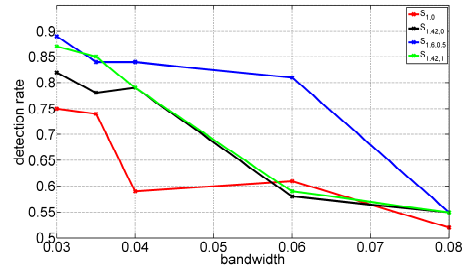
Figure 3: Detection rate ground truth

Fig. 4 illustrates the pixel-wise detection rate for k -means and mean-shift for a varying number of clusters, bandwidth, and BRDF modification factor $S_{n,m}$, as described in Eq. (6). The red curve is equivalent to a direct BRDF segmentation (spectral mean free) without any BRDF modification. Both segmentation algorithms, especially mean-shift, benefit from the reflection model modification.

A detailed image of the best segmentation result yielded by mean-shift using $S_{1.6,0.5}$ is shown in Fig. 5. The segmentation error in the lower left part of the cup can be attributed to slightly overexposed input data originating from spurious illumination from interreflections with the experimental environment. The correct separation of the three reddish regions 4, 5 and 8 (see Fig. 4(a)) is especially challenging, due to the fact that the corresponding spectra are very similar and need to be distinguished based on merely 8 spectral measurements. Additionally, an oversegmentation of the orange cluster ③ can be observed. A toleration of the oversegmentation leads to a segmentation result with a detection rate of $D \approx 95\%$ as shown in Fig. 5(c) and Fig. 5(d). Note that the segmentation is correct over nearly the whole cup, i.e. even regions with steep surface gradients (cup borders), whose measurements are typically hard to obtain due to rapidly changing incidence and viewing angles, have been segmented correctly.



(a) k -means detection rate over number of clusters



(b) mean-shift detection rate over bandwidth

Figure 4: Comparison of the correct segmentation rate between k -means and mean-shift for abrupt cluster transitions. Each curve (red, black, blue, green) corresponds to a different BRDF modification factor S .

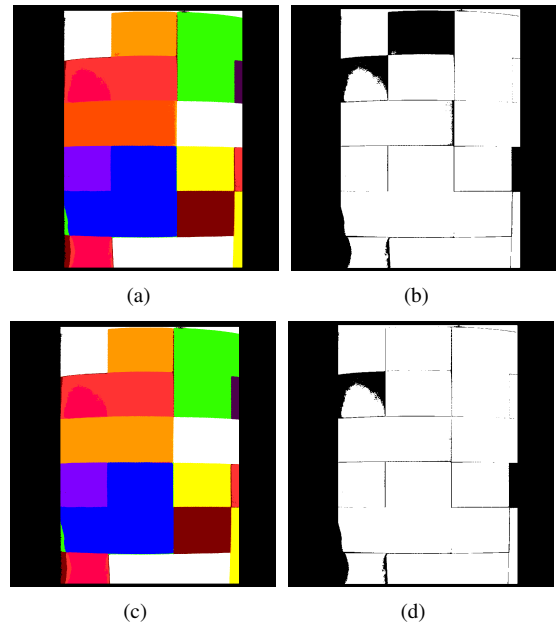


Figure 5: Segmentation result for $S_{1.6,0.5}$. (a) Best mean-shift segmentation result without further treatment. (b) Correct detected areas (detection rate: 88.45 %). (c) Best mean-shift segmentation result with toleration of the oversegmented orange cluster. (d) Correct detected areas (detection rate: 94.79 %). Note the oversegmentation of the orange cluster ((a) and (b)), which can be neglected ((c) and (d)). Note that areas at the border of the cup with steep surface gradients have been segmented correctly.

The measured BRDF can now be approximated by a Phong and/or a Cook-Torrance BRDF model for each

determined segmented cluster. Using the obtained BRDF parameters allows for a determination of a BRDF value for every half polar observation angle $\vartheta_o \in [-90^\circ, \dots, 90^\circ] \subset \mathbb{R}$, for each incident light angle $\vartheta_i \in [-90^\circ, \dots, 90^\circ] \subset \mathbb{R}$, and for each pixel belonging to the cluster. In our case, we analyze a pixel of a BRDF at 600 nm wavelength, which is located closely to a specular highlight. This pixel corresponds to cluster ①, which is generated by a mean-shift clustering of $S_{1.42,1}$ and visualized in Fig. 6(a) and Fig. 6(b).

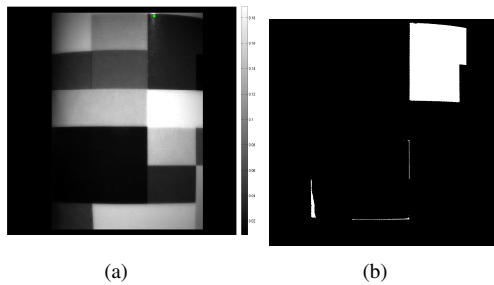


Figure 6: Data point and corresponding clusters. (a) Measured BRDF at a wavelength of 600 nm. Observed point near measured specular maximum is marked green. (b) Segmented cluster including observed data point (mean-shift approach with $S_{1.42,1}$).

Fig. 7, Fig. 8 and Fig. 9 show the results of BRDF estimation for the Phong model (single lobe as well as lobe+spike variant) and the Cook-Torrance model (lobe+spike). Note that the range of incidence and viewing angles available for BRDF estimation is narrow but the estimation is robust due to the large number of pixels involved. The obtained BRDF may become inaccurate for angles a long way off from the underlying samples used during the parameter estimation stage, but model usage far off the data basis should be avoided anyway.

An important observation is that the BRDF decreases for large absolute viewing angles $|\vartheta_o| \geq 60^\circ$. This effect has also been observed with other objects and scene settings. Relating this effect to the varying specular part of reflection does not seem very realistic due the fact that this decrease occurs at angles far away from the ideal specular reflection angle $\vartheta_{rv} = 0^\circ$. In fact, a non-ideal diffuse reflection for grazing viewing angles is a more plausible explanation. [ON94] introduce a more detailed and complex diffuse reflectance term, but they describe an increased reflectance, which is contrary to our observations. The reflectance in this work better corresponds with a “diffuse fall-off” described e.g. by [DRS08]. This effect models decreasing reflectance for larger angles of observance and is partially taken into account by [LFTG97].

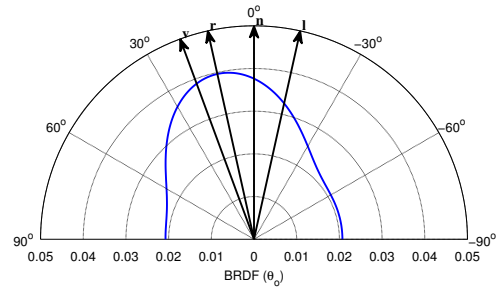


Figure 7: Phong fit (lobe).

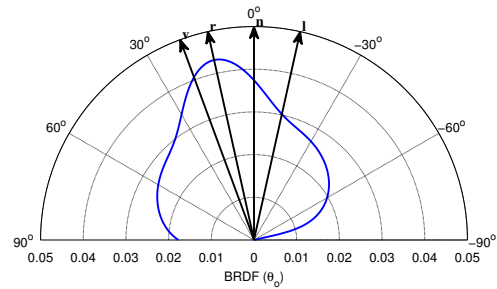


Figure 8: Phong fit (lobe+spike).

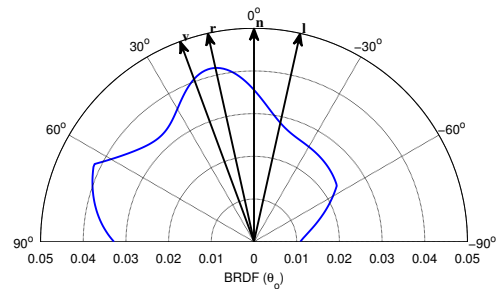
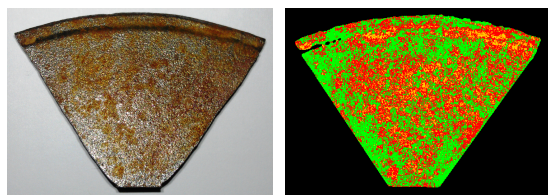


Figure 9: Cook-Torrance fit (lobe+spike).

5.2 Object with gradual region boundaries

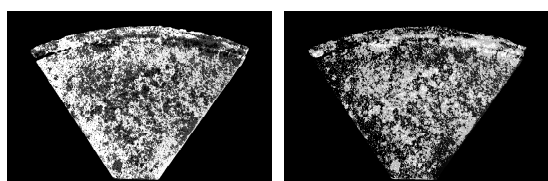
Initially, we performed a k -means segmentation (see Fig. 10) of the oxidized iron object without any modifications of the BRDF, i.e. we used $S_{(1,1)}$. To obtain a robust segmentation result without the influence of spurious effects like specular reflections and other illumination inhomogeneities, we have chosen a planar arrangement of the object with high values of the specular angle ϑ_{rv} in order to minimize the occurrence of specular reflections. However, due to the roughness of the surface there is an inevitable amount of small surface parts that reflect light specularly into the camera. These have been taken into account by a k -means segmentation with 3 clusters, i.e. iron, oxidized iron, and a “garbage” class that collects outliers. Based on 10 manually chosen reference data points for iron and rust (as required for the linear unmixing approach described in Section 4.4), an abundance distribution of both materials can be estimated as shown in Fig. 11(a) and Fig. 11(b). On the basis of this distribution and an



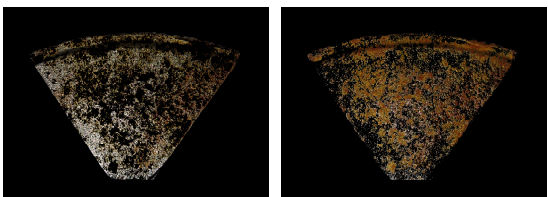
(a) Probed oxidized iron object. (b) Segmentation with k -means (3 clusters).

Figure 10: Visual comparison between original and segmented clusters. A third class (“garbage class”) besides iron and rust has been introduced to collect outliers.

image registration approach that relates the color image Fig. 10(a) with the measurement data Fig. 11(a) and Fig. 11(b), a mask of the two materials can be generated as shown in Fig. 11(c) and Fig. 11(d).



(a) Generated abundance distribution of iron. (b) Generated abundance distribution of rust.



(c) Areas with at least 60% iron. (d) Areas with at least 60% rust.

Figure 11: Linear unmixing results.

6 SUMMARY AND CONCLUSION

We have presented an approach for the acquisition and segmentation of spectral BRDF data. The data are recorded by relating object intensity data and reflectance standard data, which then directly provides BRDF measurements. We used those data to show that it is possible to obtain accurate segmentation results and endmember abundance estimates for objects with rapidly and with gradually changing materials, even if only a single light source position and a single viewpoint is available. The object surface with abrupt region boundaries has been classified with an accuracy of 88.5% (94.8% with some oversegmentation tolerance). The application of linear spectral unmixing to the separation of iron from rust for the gradually changing material has provided a qualitatively realistic result.

In summary, we thus note that it is possible to determine object regions correctly even if only a single light source and a single viewpoint are available, which is

very important for e.g. industrial measurement setups. We have observed some limitation when using raw measured depth data, which has been found to be too noisy for correct incidence and viewing angle determination. This problem has been solved by using a model-based cylindrical and plane fit for the respective objects. Using laser range scanners with higher depth and lateral resolution can overcome that limitation easily.

The Phong and Cook-Torrance BRDF models were then fitted to the previously segmented cluster data. Due to the sparse data input (limited range of incidence and viewing directions), some uncertainties in determining a hemispherical BRDF remain, especially for obliquely viewed surface parts. Additionally, the obtained BRDF tends to exhibit lower reflectance values for large observation angles $|\vartheta_o| \geq 60^\circ$, which can be explained by a “diffuse fall-off” [DRS08]. This issue requires considerable attention in further research, since it requires a phenomenological (rather than empirical) adjustment of the reflectance model.

7 REFERENCES

- [CM02] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.
- [CT81] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, 15(3):307 – 316, 1981.
- [DRS08] J. Dorsey, H. Rushmeier, and F. Sillion. *Digital Modeling of Material Appearance*. Morgan Kaufmann, 2008.
- [Ebe08] D. Eberly. Fitting 3d data with a cylinder. *online* <http://www.geometrictools.com/Documentation/CylinderFitting.pdf>, 1:1–4, 2008.
- [Fin06] B. Finkston. Mean Shift Clustering. *online* <http://www.mathworks.com/matlabcentral/fileexchange/10161-mean-shift-clustering>, 2006.
- [HB86] B. K. P. Horn and Brooks. The variational approach to shape from shading. *Computer Vision, Graphics and Image Processing*, 33:174–208, 1986.
- [HW12] S. Herbot and C. Wöhler. Self-consistent 3d surface reconstruction and reflectance model estimation of metallic surfaces. *VISAPP’2012*, pages 1–8, 2012.
- [KM02] N. Keshava and J. F. Mustard. Spectral unmixing. *IEEE Signal Processing Magazine*, January 2002:44–57, 2002.
- [LFTG97] E. P. F. Laforune, S.-C. Foo, K. E. Torrance, and D. P. Greenberg. Non-linear approximation

- of reflectance functions. *SIGGRAPH'97*, pages 117–126, 1997.
- [LHW12] M. Lenocho, S. Herbot, and C. Wöhler. Robust and accurate light source calibration using a diffuse spherical calibration object. *Oldenburger 3D Tage 2012*, 11:1–8, 2012.
- [LKG⁺03] H. P. A. Lensch, J. Kautz, M. Goesele, W. Heidrich, and H.-P. Seidel. Image-based reconstruction of spatial appearance and geometric detail. *ACM Transactions on Graphics*, 22(2):234–257, 2003.
- [LMM98] G. Lukacs, R. Martin, and D. Marshall. Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. *ECCV '98*, 1:671–686, 1998.
- [Mar09] S. Marsland. *Machine Learning: An Algorithmic Perspective*. Chapman & Hall/CRC Machine, 2009.
- [NRN03] S. G. Narasimhan, V. Ramesh, and S. K. Nayar. A class of photometric invariants: Separating material from shape and illumination. *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, pages 1387–, 2003.
- [ON94] M. Oren and S. K. Nayar. Generalization of lambert's reflectance model. *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1994)*, pages 239–246, 1994.
- [Pho75] B. T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–17, 1975.
- [Sha85] S. A. Shafer. Using color to separate reflection components. *Color Research and Application*, 10(4):210–218, 1985.
- [Tom02] S. Tominaga. Region segmentation by multi-spectral imaging. *Fifth IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 238–242, 2002.
- [Woo80] R. J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):139–144, 1980.

8 APPENDIX

Fig. 12 and Fig. 13 show image data for four exemplary wavelengths ($\lambda = 450\text{nm}, 550\text{nm}, 650\text{nm}, 750\text{nm}$). Note, that the pixels that correspond to a certain color / material have a designated intensity characteristic. The mean shift and k -means approaches described in the paper analyze that characteristic and group similar characteristics. It can be observed that e.g. rust has a lower 450nm component than iron (i.e. less blue), but contains a more intense 650nm and 750nm component (more red). These relations exist over the whole dataset and thus allow the application of the unmixing approach.

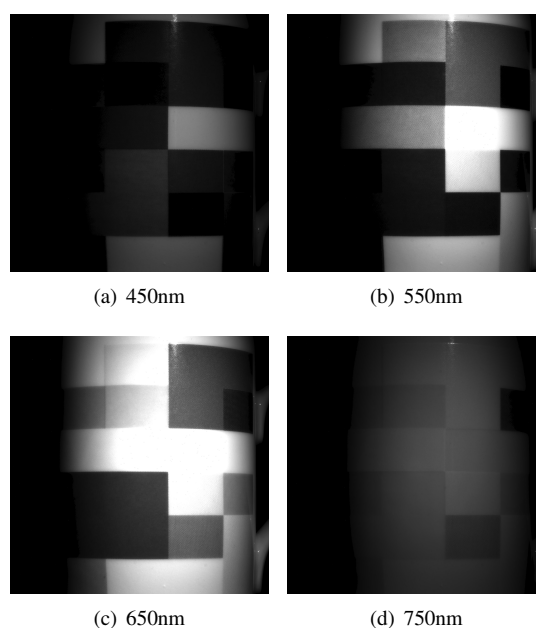


Figure 12: Camera image examples from the “cup” dataset. All images have been converted to the same reference intensity to facilitate image comparisons.

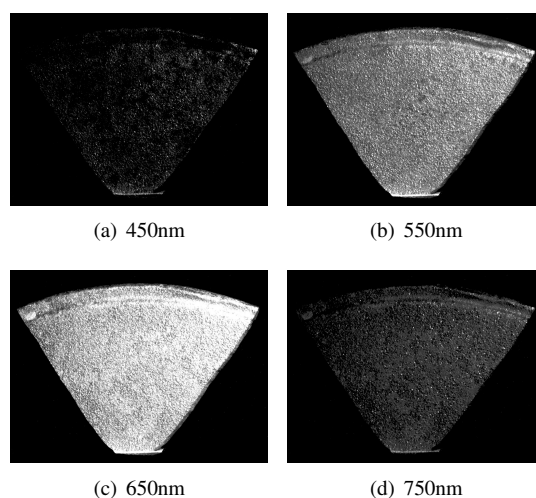


Figure 13: Camera image examples from the “triangle” dataset. All images have been converted to the same reference intensity to facilitate image comparisons.

Semi-Automatic Synthesis of Videos of Performers Appearing to Play User-Specified Music

Tomohiro Yamamoto

UEC
yamamoto@onailab.com

Makoto Okabe

UEC/JST PRESTO
m.o@acm.org

Yusuke Hijikata

UEC
hijikata@onailab.com

Rikio Onai

UEC
onai@cs.uec.ac.jp

2-11, Fujimicho, Chofu, 182-0033, Tokyo, Japan

ABSTRACT

We propose a method to synthesize the video of a user-specified band music, in which the performers appear to play it nicely. Given the music and videos of the band members as inputs, our system synthesizes the resulting video by semi-automatically cutting and concatenating the videos temporarily so that these synchronize to the music. To compute the synchronization between music and video, we analyze the timings of the musical notes of them, which we estimate from the audio signals by applying techniques including short-time Fourier transform (STFT), image processing, and sound source separation. Our video retrieval technique then uses the estimated timings of musical notes as the feature vector. To efficiently retrieve a part of the video that matches to a part of the music, we develop a novel feature matching technique more suitable for our feature vector than dynamic-time warping (DTW) algorithm. The output of our system is the project file of Adobe After Effects, on which the user can further refine the result interactively. In our experiment, we recorded videos of performances of playing the violin, piano, guitar, bass and drums. Each video is recorded independently for each instrument. We demonstrate that our system helps the non-expert performers who cannot play the music well to synthesize its performance videos. We also present that, given an arbitrary music as input, our system can synthesize its performance video by semi-automatically cutting and pasting existing videos.

Keywords

Video Synthesis Music Analysis Multimedia

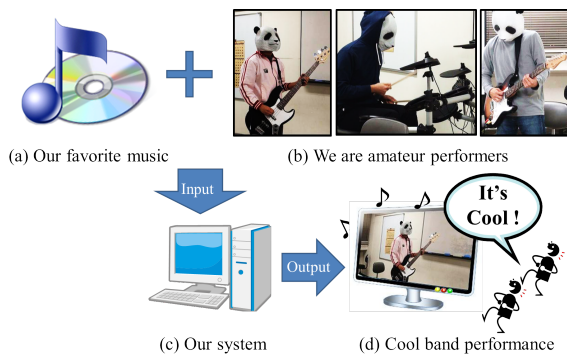


Figure 1: We want to synthesize the video of our favorite music (a) played by our band members. We have recorded the performance videos of our band members (b), but they were not perfect. So, we developed a system (c) that takes the music and the videos as inputs, and synthesizes the video composite (d) in which all the members appear to play the music nicely.

1 INTRODUCTION

Synchronization between sound and footage is an important task when producing a high-quality movie. For this task, the designers must spend a lot of time 1) to add sound effects or music to a silent footage or 2) to

edit the footage so that it synchronizes with the given sounds or music. In this paper, we focus on this second demand, i.e., we develop a system to help the designer, who creates a musical performance video, to efficiently synthesize it.

Today, more and more people have become interested in synthesizing an original video using their favorite music. Especially, in video-sharing services such as YouTube, we can find a lot of music videos created by the professional and amateur designers, which a lot of people enjoy every day. Note that watching the music video is the totally different experience from only listening to the music. For example, we can tend to pay more attention to the sounds of the currently watching instrument: watching the bass performance allows us to more easily capture the low-pitched bass sounds. It is also reported that the movie affects how we, audiences, feel the rhythm and moods of the music [10].

As related researches, there are several methods that synthesize videos synchronizing to user-specified music, e.g., the summarization of home videos based on a user-specified music [4], the creation of the slideshow video of photographs [6], or the dance performance videos using 3D human models [5, 11] or the 2D performance videos [8]. However, since all of these

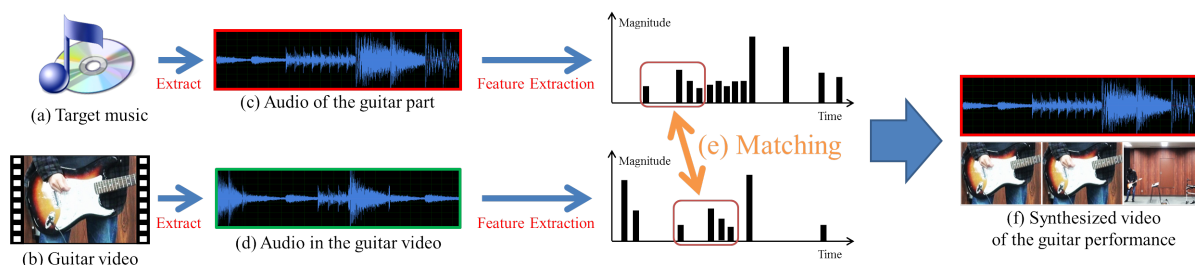


Figure 2: The overview of synthesizing a performance video of a single instrument (Guitar, here).

approaches analyze only the tempos and moods of the input music, we cannot apply these techniques to our problem: since we want to synthesize a performance video in which the performer looks like playing the music, we require finer synchronization.

Given a user-specified music and videos of performers, our method semi-automatically synthesizes the music video. We usually record the video for each instrument independently and then create the video database. We then estimate the timings of the musical notes by analyzing the audio signals of the music and video database. Using the estimated timings as the feature vector, our system retrieves the candidates of footages for each part of the input music. The output is the project file of Adobe After Effect that has multiple candidates, in which the user can further edit the result.

We asked the two types of performers to join in our experiments. In the first experiment, we asked amateur performers who can play each instrument well. We recorded the performances of playing viola, drums, and bass. In the second experiment, our performers are novices, i.e., who didn't know how to play each instrument at first: we then asked them to practice each instrument for one hour. In each case, we demonstrate that our system can synthesize a reasonable performance video in short editing times.

2 SYSTEM OVERVIEW

Our system consists of 2 parts: the first part synthesizes a solo performance video of a single instrument (Fig. 2), and the second part synthesizes the band performance video by mixing all the synthesized solo videos (Fig. 3). In Fig. 2, The user begins to synthesize a solo performance video by specifying a favorite music (Fig. 2-a). If the input music includes multiple instruments, we first separate it into solo audio signals using sound source separation technique (Sec. 3.2).

Here, we show the guitar signal of the input music (Fig. 2-c), and the video of playing the guitar (Fig. 2-d). We then apply our technique of extracting the timings of musical notes (Figs. 2-e and f). In each feature vector, the peak position corresponds to the estimated timing of a musical note, and the distance from it to the next peak corresponds to the duration of the note.

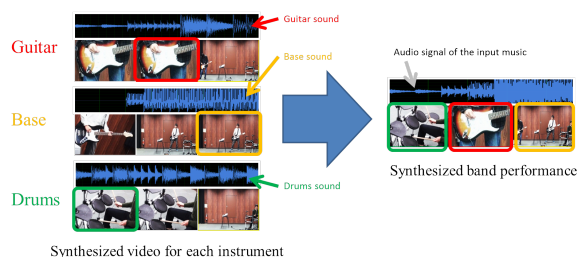


Figure 3: To synthesize the band performance consisting of multiple instruments, our system selects the adequate part from each solo performance video and concatenate them. Here, we select red, orange, and green parts and concatenate them.

The system then computes the part-by-part matching between the feature vectors of the input music and the video (Fig. 2-e). Finally, we synthesize the solo performance video by concatenating the videos based on the matches (Fig. 2-f). After synthesizing solo performance videos for each instrument, our second part synthesizes the band performance video by further cutting and concatenating them (Fig. 3).

3 FEATURE EXTRACTION

In this section, we describe the feature extraction, i.e., the process of estimating the timings of musical notes from the audio signal of a solo performance (Sec. 3.1). When the audio signal has sounds of multiple instruments, the sound source separation is required, which separates an audio signal of multiple instruments into audio signals of each single instrument (Sec. 3.2).

3.1 Feature Extraction on Solo Performance

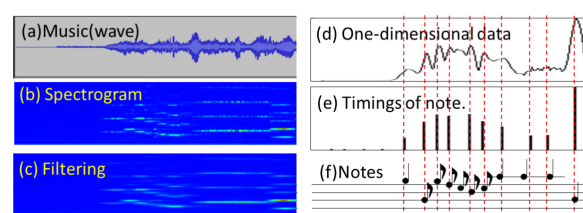


Figure 4: The extraction of feature vector.

To extract the feature vector from an audio signal, we develop a simple technique of onset detection. Our idea is similar to [1]. We extract the feature vector from input music and video database.

To estimate the timings of the musical notes, our method begins to compute short time Fourier transform (STFT) to the audio signal (Fig. 4-a), and obtains the spectrogram (Fig. 4-b). Since the spectrogram is usually noisy, we smooth it out but preserve strong edges that correspond to the beginnings and endings of each musical note. For this purpose, we apply a horizontal bilateral filter [14] to obtain the smoothed spectrogram (Fig. 4-c). We differentiate the smoothed spectrogram horizontally to extract the beginning and end of each musical note. We then integrate the spectrogram vertically to obtain one dimensional (1D) signal (Fig. 4-d). Finally, we extract the feature vector by finding local maxima of the 1D signal (Fig. 4-e). We show the score of this audio signal that the performer actually played in Fig. 4-f. Note that the peaks match the timings of the score.

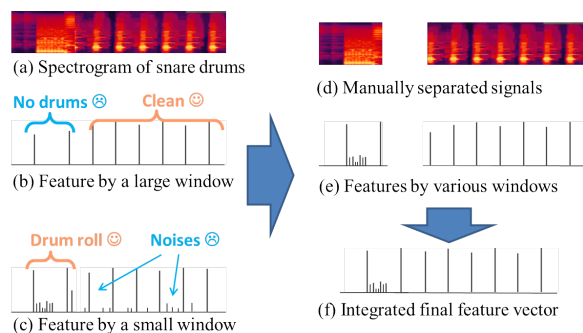


Figure 5: The technique to extract the feature vector using multiple bilateral filters with various window sizes.

To extract high quality feature vectors, we must choose an adequate window size for the bilateral filter. For example, Fig. 5 shows an audio signal of drums, which starts by the high-frequency drum roll. A large window produces clean peaks but makes the peaks around the drum roll disappear. On the other hand, a small window produces the peaks of the drum roll well but also too many small peaks that are noises. So, we need to change the window size adaptively through the signal. However, since it was difficult to implement such a smart adaptive filter, we decided to manually cut the signal into segments based on the observed frequencies, and apply the adequate bilateral filter to each of them. This process increases the user's burden but is important to create the high quality feature vector.

We also use the sound volume values as our additional feature vector. We use volume information to detect whether the instrument is being played or silent: if it is silent, we don't assign any footage of the instrument to that part. Our feature vector of the volume is binary, i.e., 1 (there is sounds) or 0 (silent).

We also use the changes of pitch values as our additional feature vector. To estimate the pitches of an audio signal, we use STRAIGHT [7] to estimate the fundamental frequency (F0). Since we are interested not in the absolute pitch values but only in the relative changes of pitch values, we use the differentiation of F0 as our feature vector. This feature vector helps to synthesize the reasonable performance video by preventing from assigning a footage of descending pitch to that part of ascending pitch.

3.2 Sound Source Separation

The algorithm of feature extraction described above works well for an audio signal of a single instrument. However, it is often difficult to prepare the audio track for each instrument independently. For example, the drum set that we used to record the drummer's performance gives us only the mixture of sounds of four drums, i.e., snare drum, bass drum, cymbals, and hi-hats. In such a case, we want to separate the audio track for each instrument. For this purpose, we use the semi-automatic sound source separation technique. Fig. 6 shows an example of the separation of the drum audio signal.

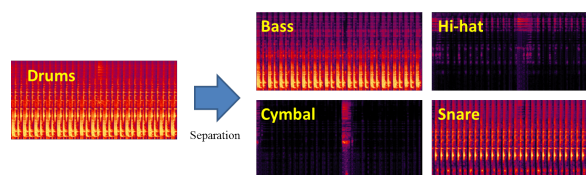


Figure 6: Sound source separation of drums.

We use the probabilistic latent component analysis (pLCA) [12], which is developed based on pLSA algorithm [2]. pLCA is the algorithm for non-negative matrix factorization that separates a spectrogram $S(t, f)$ as follows:

$$S(t, f) = \sum_i^N B_i(t, f), \quad (1)$$

where $B_i(t, f)$ is non-negative, t represents the time, f represents the frequency, and N is the number of the bases. Fig. 7 shows an example of the separation of an audio signal: the original spectrogram having the violin and drums sounds is separated into the twelve bases at first. We then manually classify them into two classes in this case, i.e., violin class and drums class. For the violin class, the user's selection is visualized as the green boxes. We sum these bases to obtain the spectrogram of the violin. These separated sounds are demonstrated in the supplementary video.

4 VIDEO RETRIEVAL BY FEATURE MATCHING

For each part of the input music, we retrieve a footage so that the feature vector of the part of the input music

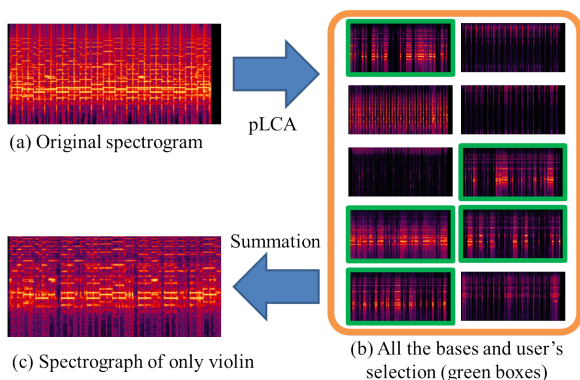


Figure 7: Sound source separation by pLCA.

matches to the feature vector of the footage. However, it is unusual that a pair of feature vectors match exactly, and if we consider only such exact matches, we cannot retrieve any footage for most parts of the input music. So, we consider not only exact matches but also similar matches. Fig. 9 shows some examples: each of red, green, and blue boxes show the match of feature vectors. The peaks of the same number are matched, but the timings of them are slightly different. We complement these differences by changing the playback speed of the footage in the video synthesis process.

4.1 Our Algorithm

As a method for matching the signal while stretching, dynamic time warping (DTW) is well-known [9]. DTW works well for computing similarities between smooth, continuous signals, but does not work for our case, i.e., each signal consists of discontinuous peaks. More precisely, DTW works well for our case, only when the number of peaks is the same between the comparing signals; however, since the method of feature extraction described above is not always perfect, it often produces noises. Fig. 8 shows an example. Signals A and B share two peaks of similar magnitudes, but signal A has a small peak as noise between them. When we compute the warping path of DTW, it becomes as the path of red circles, i.e., it makes the second peak of signal B match to the noisy peak. Based on this observation, we concluded that DTW is not suitable for the comparison of peaky signals.

To solve the problem, we propose a simple technique to efficiently retrieve a footage, allowing the temporal stretching. Our idea is inspired by the RANSAC algorithm [3] used in many computer vision applications, which efficiently finds the set of matches that are consistently explained by a transformation.

Fig. 9 shows how our algorithm finds a local match. The red line of the first step shows 1-to-1 match of peaks: In this step, we find such a 1-to-1 match by simply comparing the magnitudes of peaks using a threshold parameter, α . In the second step, we check the

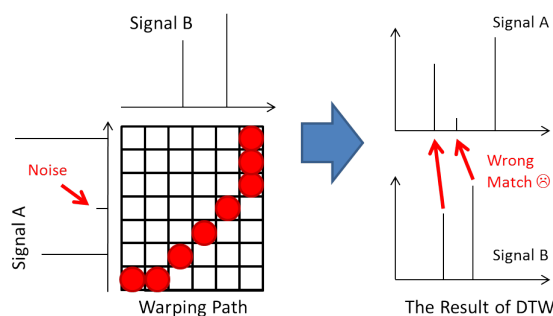


Figure 8: DTW algorithm is not suitable for our case, where the signal consists of peaks.

neighboring peaks: here, peaks '3' and 'C' can be a match, because the distance between '3' and '1' and the distance between 'C' and 'A' are similar. The same thing can be said for the match of '2' and 'B'. In the third step, we further check the neighboring peaks: here, peaks '4' and 'D' can be a match because the distances are similar. However, peaks '5' and 'E' cannot be a match: the distance between '5' and '3' are too smaller than the distance between 'E' and 'C'. In the next step, we further check the neighbors toward right-hand side, but we don't check the neighbors toward left-hand side anymore. We continue this process until the difference of the distances becomes larger than a threshold parameter, β . We apply this algorithm to all the possible 1-to-1 matches between the input music and the video database.

Our algorithm solves the weakness of DTW, and inherits the strength of DTW at the same time. The weakness of DTW is that it is too sensitive to noises as shown in Fig. 8. However, in our algorithm, we can prevent a healthy peak from matching to such a small noisy peak by appropriately specifying the threshold α . On the other hand, the strength of DTW is that it can take the temporal stretch into account, which can be achieved by appropriately specifying the threshold β . Fig. 9-right shows the three resulting local matches of blue, red, and green boxes. For example, in the blue match, the distances of features of the input music are larger than the distances of features of the video database, which means that we can fit the video by making the playback speed slower. In the green match, we make the playback speed faster and the video would fit to the part of the input music.

5 SYNTHESIS OF BAND PERFORMANCE VIDEO

Given the input music and a set of videos of playing each instrument, the methods described above extract the feature vectors, retrieve an appropriate footage and assign it to each part of the input music. Each part of the input music usually has multiple candidates of footages. Our system saves this result of retrieval and assignment

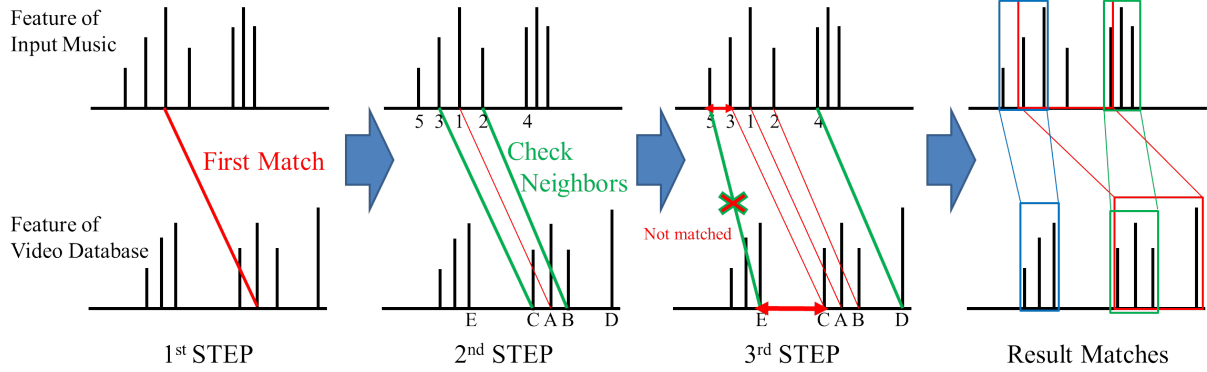


Figure 9: Our algorithm to find local matches.

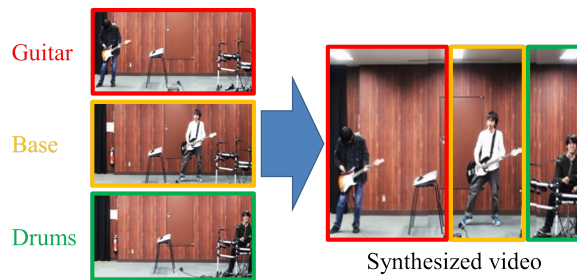


Figure 10: Masking and lapping the solo performance videos to synthesize the band performance video.

as the project file of Adobe After Effects: the user can interactively select the best set of footages from the candidates and render the final movie. The temporal stretch of each footage is easily achieved by the time remapping function of Adobe After Effects: when generating the project file, our system automatically specifying the parameters of this function for each candidate footage.

This editing process is usually as shown in Fig. 3: since our method assumes that each video is recorded for each instrument independently, we cannot synthesize a scene where all the band members appear at the same time. One idea to synthesize such a scene is to record each performer also from a fixed camera as shown in Fig. 10. Each video has just a solo performance, but we can synthesize all the band members by spatially masking and lapping all the videos into the resulting video.

5.1 Automation of Candidate Selection

Manually selecting the best footage for all the parts of the input music is often hard work and time-consuming. To reduce this user's burden, we develop an automatic method to support this process. We consider this as the problem of label assignment for each video frame: each label corresponds to each candidate footage, i.e., the number of labels L is the same as the number of candidate footages. We formulate this as Markov Random Field (MRF) as follows:

$$\operatorname{argmin}_i E = \sum_p V_p(l_i) + \lambda \sum_{p,q} W_{p,q}(l_i, l_j), \quad (2)$$

where labels of l_i and l_j are assigned to the neighboring frames of p and q . V_p is the data term, and $W_{p,q}$ is the smoothness term. $V_p(l_i)$ is defined by the Euclidean distance between the feature vector of the input music at p frame and the feature vector of the candidate footage l_i . $W_{p,q}(i, j)$ defines the cost of transition from footage l_i at p -th frame to footage l_j at q -th frame. We can solve the energy minimization using the graph-cut algorithm with α - β swap [13].

Fig. 11 shows an example of the energy minimization. Here are five candidate footages, A, B, C, D, and E. Here are nine frames. For the 1st and 2nd frames, candidate A is assigned, because the data term of A, 0.1 is smaller than the data term of B, 0.2. For the same reason, candidate E is assigned to the 8th and 9th frames. On the other hand, for the 3rd frame, candidate B who has smaller value, 0.2, is not assigned, but candidate C is assigned. This is because, if we assign B here, it produces a short footage whose duration is just one frame at the 3rd frame: the resulting video that has many such short footages looks annoying because of many scene transitions. We use the smoothness term to avoid this, i.e., $W_{p,q}$ is set to a large value for the transition that might cause too short frame. More concretely, $W_{2,3}(A, B)$ is set to a large value, here.

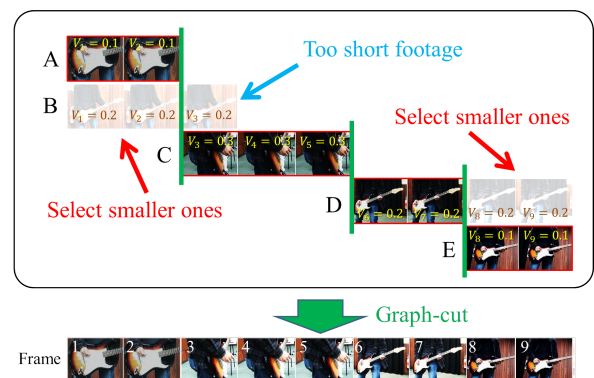


Figure 11: The energy minimization of MRF automatically selects the reasonable candidates. Here are five candidates, A, B, C, D, and E, and here are nine frames.

6 RESULTS

We present that our system can synthesize the video of performers appearing to play user-specified music. In the first experiment, we present that our system can synthesize the video of an arbitrary music using the video database of amateur performers. In the second experiment, we try to synthesize the music video using videos of performers who cannot play the instruments. Finally, we perform subjective evaluations to analyze the usability of our system and the quality of the synthesized videos.

6.1 Experiment 1

As the input music, we selected two music. One is “Let it be” of The Beatles, and the other is “Etupirka” of Taro Hakase. As for “Let it be” played with a violin, a bass, and drums, we could prepare the wave files for each instrument in advance. As for “Etupirka” played with a violin and drums, since we could prepare only the original wave file, we applied semi-automatic sound source separation technique to separate it into the violin part and the drums part. As for the video database, we asked the amateur performers of viola, bass, and drums to play each instrument. We did not specify what they should play but asked to play the instrument freely, i.e., any music randomly. We recorded their performance for one hour using four cameras at the same time. These cameras are set at the different viewpoints.

The supplementary video has the results. (Please use an appropriate video player to watch them: in our environment, Windows Media Player did not play our results with nice synchronization, i.e., there were significant time shift between the audio and the footages. Quick-Time player was much better.) Note that each sound synchronizes with the movie, especially, drum sounds do.

The result of “Etupirka” demonstrates the limitation of our technique: we can find the scenes where the performer moves but there is no sound (or the performer does not move but there are sounds). This is caused by the failure of the sound source separation. Fig. 13 shows the feature vectors in “Let it be” and “Etupirka”. “Let it be” was actually good, but “Etupirka” is more noisy. Sound source separation is a difficult problem. Our manual selection of pLCA bases makes it possible to obtain relatively reasonable separation results, but still causes such noises. To synthesize a high quality music video, if we could prepare the audio track for each instrument independently, it would be the best.

Since these results are synthesized using graph-cut, we did not spend much time to synthesize the performance video: we took about 15 minutes to synthesize each of these videos.

6.2 Experiment 2

We prepared the input music “untitled” by an amateur composer for this experiment: this music is played with the guitar, the bass, the piano, and the drums. We asked four students from the computer science department to be the performers: these students have had little experience of the instruments. We allowed them to practice each instrument for one hour, and recorded their performance using two cameras: one camera was hand-held and moving, and the other was fixed.

The supplementary video has the result. While these performers were novices and just pretending to playing the instrument, our system can synthesize the performance video by retrieving footages and assigning them to the input music. Also, as described above, we synthesized the scenes of all the band members by mixing the solo performance videos recorded from the fixed camera.

The guitar and bass performers always strike the same string, and the fingers of the left hand do not move much. As the result, if the change of the pitch of input music is significant or the audience has knowledge about the instrument, the unnaturalness of the synthesized video would become noticeable.

To synthesize this result, we did not use graph-cut to synthesize more high-quality video. We further spend time to edit camera motions digitally. We took 3 hours to synthesize this band performance video.

6.3 Subjective Evaluation

We performed a subjective evaluation of the quality of the synthesized video. We prepared six types of the videos:

- v1: video of "Let it be" synthesized using graph-cut
- v2: video of "Let it be" synthesized without graph-cut
- v3: video of "Etupirka" synthesized using graph-cut
- v4: video of "Etupirka" synthesized without graph-cut
- v5: video of "untitled" synthesized without graph-cut
- v6: video of "untitled" synthesized using After Effects

We asked the 15 students of the computer science department to evaluate “how unnatural each video looks”. Fig. 15 is the result of this user study, where the score 0 means there is no unnatural scene, and the score 6 means that the video is full of unnatural scenes.

v2 and v4 got smaller scores than v1 and v3. This shows that the quality of the video produced by automatic candidate selection tends to be lower. However, this result also demonstrates that the quality of the graph-cut is not bad. Especially, the score of v4 is comparable to that of



Figure 12: The frames from the “Let it be” and “Etupirka” videos.

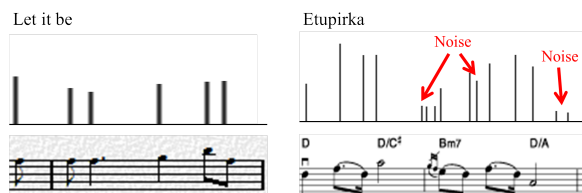


Figure 13: The example of the feature vectors.

v3. We believe the additional manual edit on the result produced by the graph-cut would improve the quality.

v1 got the worst score. The score of v2 is not good, either. The main reason was the drums: in the video of the drum player, he beats the tom. However, since the sound source separation between tom and snare drum was difficult, we treat these sounds as the same. As the result, the evaluators realized this fact, and they assigned the bad score to this video. In v2, since such unnatural scenes could be removed manually, the score is better.

The interesting fact was that the score is affected by the evaluator’s experience of each instrument a lot: several evaluators who had experiences to play the guitar or drums tend to realize the unnaturalness of these instruments in the video. On the other hand, no evaluator had experience to play the violin, and there was few comments about the unnaturalness of the violin. In v3 and v4, the violin is the main instrument, and the drums are simple, which we believe is the reason about why the score of them are better than v1 and v2.

v5 and v6 are for comparison between our system and standard video editing software. We chose Adobe After Effects as the video editing software. To make v5, we manually selected the candidate footages on Adobe After Effects. To make v6, we used only Adobe After Effects, which was completely manual work: we manually search for an adequate video from the database, and edit it. For both videos, we did our best to synthesize as the high quality video as possible. The difference of these scores is very small, which means the qualities of the resulting videos are almost the same. This fact proves that our system can synthesize the video whose quality is similar to the video synthesized completely manually. The performers in these videos had almost no knowledge about how to play each instrument. Most of the evaluators realized the unnaturalness of the performance, e.g., guitar and bass players’ wrong motions. It is difficult to synthesize the perfect music video of these band members: if we want to synthesize it, we

must ask them to practice the instruments hard. This is the other limitation of our method.

Finally, we measured and compared the time required to synthesize a short, solo performance video by our system and After Effects. We asked three test users who are the students of computer science department to synthesize a short video. Since all of the test users were not familiar with video editing, we taught how to use our system and After Effects to each test user, and gave 10 minutes to practice it respectively. We then ask them to synthesize a short video whose duration is several seconds consisting of a single footage. Fig. 16 is the average time spent to finish the task: our system took 59 ± 10 seconds, and After Effects took 578 ± 157 seconds.

When using After Effects, the user had to check the long video, retrieve an adequate part of it, and then manually assigns it the part of the music: when the assignment process, the manual temporal stretching of the footage is also required. On the other hand, since our system shows the candidate footages automatically, the burden of the user is dramatically reduced: all the user had to do is to check the candidate footages and select one of them. Also, when using After Effects, the qualities of the resulting videos are different from subject to subject. Using our system, the qualities are almost the same.

Through these subjective evaluations and user studies, we have shown that our system can synthesize music videos whose qualities are comparable to the videos manually edited using the existing software. We also demonstrated that the user’s burden to create the similar quality video is much less than the existing software. Also, our system is simple and can be used even by a novice user, after 10 minutes tutorial.

7 CONCLUSION AND FUTURE WORK

In this paper, we proposed a method to synthesize the video of performers appearing to play user-specified music. Technically, our method consists of feature extraction, semi-automatic sound source separation, video retrieval based on local feature matching, and automatic candidate selection by graph-cut algorithm. Through the experiments, we have shown that our system can synthesize the music video with the comparable quality but less burden than the standard video editing software.



Figure 14: The frames from the “untitled” videos.

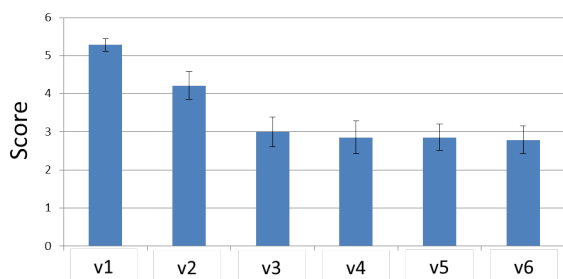


Figure 15: The evaluation of unnaturalness of the six synthesized videos.

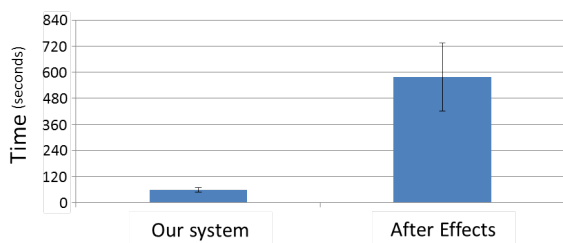


Figure 16: The time required to synthesize a short video.

In the future, we want to synthesize more high-quality video using performance videos of a professional band: currently, the band members are just amateurs or novices. We would like to try to synthesize a really professional music video and investigate how our system helps such a professional demand. The other future direction is to extend the method to take the visual information into account: currently, our synthesis method relies only on the audio information. However, by analyzing the body motion of the performers, we expect that we can exploit more semantic information for the music video synthesis.

8 ACKNOWLEDGEMENT

This work was supported by JSPS KAKENHI : Grant-in-Aid for Scientific Research (C) Grant Number 23500114. We also thank saku for providing the original music.

9 REFERENCES

- [1] Bello, J., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., Sandler, M.: A tutorial on onset detection in music signals. *IEEE SAP* pp. 1035–1047 (2005)
- [2] Brants, T., Chen, F., Tsochantaridis, I.: Topic-based document segmentation with probabilistic latent semantic analysis. In: *Proc. of CIKM*, pp. 211–218 (2002)
- [3] Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* pp. 381–395 (1981)
- [4] Foote, J., Cooper, M., Girgensohn, A.: Creating music videos using automatic media analysis. In: *Proc. of ACM Multimedia*, pp. 553–560 (2002)
- [5] Goto, M.: An audio-based real-time beat tracking system for music with or without drum-sounds
- [6] Hua, X.S., Lu, L., Zhang, H.J.: Automatically converting photographic series into video. In: *Proc. of ACM Multimedia*, pp. 708–715 (2004)
- [7] Kawahara, H., Morise, M., Takahashi, T., Nisimura, R., Irino, T., Banno, H.: Tandem-straight: A temporally stable power spectral representation for periodic signals and applications to interference-free spectrum, f0, and aperiodicity estimation. In: *Proc. of ICASSP* (2008)
- [8] Nakano, T., Murofushi, S., Goto, M., Morishima, S.: Dancereproducer: An automatic mashup music video generation system by reusing dance video clips on the web. In: *Proc. of SMC*, pp. 183–189 (2011)
- [9] Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. In: *Readings in speech recognition*, pp. 159–165 (1990)
- [10] Schutz, M., Manning, F.: Looking beyond the score: The musical role of percussionists’ ancillary gestures. In: *A journal of the Society for Music Theory* (2012)
- [11] Shiratori, T., Nakazawa, A., Ikeuchi, K.: Dancing-to-music character animation. *Comput. Graph. Forum* pp. 449–458 (2006)
- [12] Smaragdis, P., Raj, B., Shashanka, M.: Supervised and semi-supervised separation of sounds from single-channel mixtures. In: *Proc. of ICA*, pp. 414–421 (2007)
- [13] Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: *IEEE Trans. Pattern Anal. Mach. Intell.* pp. 1068–1080 (2008)
- [14] Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: *Proc. of ICCV*, pp. 839–846 (1998)

Representing Feature Location Uncertainties in Spherical Images

Bernd Krolla
DFKI - German
Research Center for
Artificial Intelligence
bernd.krolla@dfki.de

Gabriele Bleser
DFKI - German
Research Center for
Artificial Intelligence
gabriele.bleser@dfki.de

Yan Cui
DFKI - German
Research Center for
Artificial Intelligence
yan.cui@dfki.de

Didier Stricker
DFKI - German
Research Center for
Artificial Intelligence
didier.stricker@dfki.de

ABSTRACT

Pose uncertainty estimation of calibrated cameras is a common task in the field of computer vision and uses location uncertainties of image features. For spherical cameras, those uncertainties cannot be optimally described using conventional latitude-longitude representation. Increasing distortions close to the poles of the spherical coordinate system prevent a suitable description through Gaussians.

To overcome this limitation, we present a consistent location uncertainty representation for spherical image features: Our approach is based on normal vectors in Cartesian space and applicable to any kind of camera with convex projection surfaces, such as catadioptric and spherical systems. We compare its performance against latitude-longitude representation by estimating camera pose uncertainties through first order error propagation in a weighted least squares pose estimation scenario. Our experiments on synthetic and real data show that the proposed approach delivers consistent results outperforming conventional latitude-longitude representation.

Keywords

Spherical imaging, feature extraction, uncertainty visualization, camera pose optimization

1 INTRODUCTION

Spherical imaging experienced increasing attention in the recent past: Microsofts Streetside project as well as Googles Street View project [ADF⁺10] exemplify the usability of spherical images for large scale applications. The availability of omnidirectional cameras like SpheronVRs SceneCam[VR], Weiss AGs Civetta[Wei] or the Ladybug camera from Point Grey Research also corroborate the interest on spherical imaging.

Since those cameras provide the largest possible field of view (FOV), their usage avoids problems, known from perspective imaging: Visual SLAM can be performed without losing image features caused by camera rotation in combination with a limited FOV as studied by Gutierrez et al. [GRMG11]. Furthermore provide omnidirectional cameras due to the extended FOV more extractable features for spherical Structure from Motion (SfM) algorithms as proposed by Pagani et al. [PS11] and Torii et al. [TIO05].

The propagation of image feature uncertainties can augment spherical SfM and SLAM algorithms and improves for example outlier detection or surface fitting in a subsequent reconstruction process.

Since feature uncertainty description in perspective images can be handled based on Gaussian error approximation in the Cartesian coordinate system [BBS07, ZGS⁺09], this approach is not applicable for spherical images. A spherical image is hereby considered as a mapping from a given three-dimensional environment through the camera center onto a unit sphere. As this sphere is a two-dimensional manifold, two parameters are sufficient to identify any point unambiguously.

Building on that, spherical cameras typically provide images in an unwrapped form of the widely used latitude-longitude representation relying on (θ, ϕ) -coordinates with $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$ (Figure 1(b) and 2(a)).

Problem: Since the description of image feature uncertainties implies the neighborhood of the features, local distortions as they occur close to the poles of the spherical coordinate system, affect the uncertainty description. A Gaussian distributed uncertainty on a sphere is therefore not adequately describable by a single Gaussian in (θ, ϕ) -coordinates (Figure 1(a) and 1(b)). Due to the discontinuity of the longitude-coordinate at the poles, location uncertainties spreading over the image pole cannot be represented at all.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

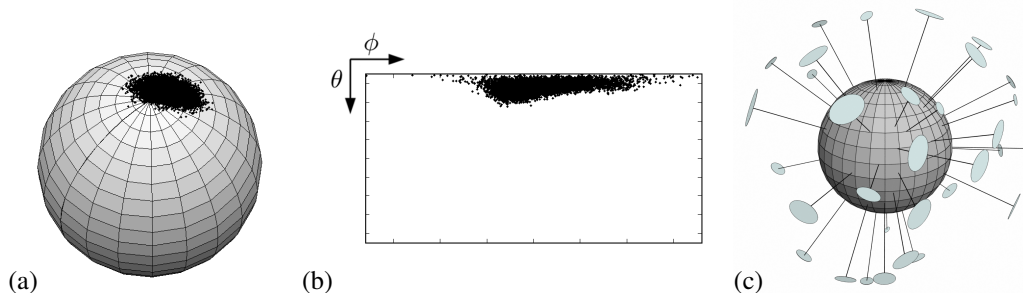


Figure 1: The Monte-Carlo based transformation from a Gaussian distribution on a sphere (a) to latitude-longitude representation (b) illustrates, that the resulting distribution cannot be properly described by a Gaussian. We propose therefore to quantify feature uncertainties through 3D uncertainties attached to normal vectors (c). Compared to latitude-longitude representation, this approach describes uncertainties consistently throughout the spherical image.

Contribution: In this work, we introduce an uncertainty representation based on normal vectors (Figure 1(c)) to overcome the previously mentioned limitations. Our proposed normal vector representation is not only limited to full spherical images. Spherical or catadioptric cameras producing a locally varying pixel-per-angle ratio as examined by Streckel and Koch [SK05] can be handled by replacing the sphere with ellipsoidal shapes.

Related work: In the field of computer vision and SfM much effort has been put into the deduction of uncertainty propagation techniques within the last years [CF05, CZZF97, DLLP11, Har98, HZ00], including approaches like first order error propagation [BBS07] or sparse grids [JU04]. Even though the listed work covers exclusively perspective imaging.

The approach taken in this work to avoid given singularities of latitude-longitude representation can to a certain extend be considered to be analogue to the concept of quaternions: Stuelpnagel shows in [Stu64] that all three-parameter representations of rotations are highly nonlinear and possess singularities in their description in three dimensional space. In order to prevent the occurrence of these singularities, the usage of quaternions introduces an additional dimension. Our usage of the normal vector exploits the same principle of introducing an additional third dimension to avoid singularities in feature uncertainty representation on a sphere.

The concept of normal vectors for position representation is furthermore not limited to spherical image feature representation but is also used in different context: Gade proposes for example a method for global navigation based on normal vectors [Gad10].

In the course of this work location uncertainties of image features are assumed to be Gaussian distributed in image coordinate system. Zeisl et al. [ZGS⁺09] show that this assumption is valid for the uncertainty characterization of SIFT [Low04] and SURF [BTVG06] de-

scriptors. For other state of the art descriptors, such as CARD [AY11], BRISK [LCS11], DAISY [TLF10] or ORB [RRKB11] the appropriate feature uncertainties can be deduced under consideration of their particular manner of feature extraction.

Outline: The extraction of feature uncertainties from spherical images is described in section 2. The following section 3 outlines the proposed uncertainty description. Section 4 contains a comparison of our method with the conventional latitude-longitude representation by performing uncertainty propagation and calculating camera pose uncertainties for different scenarios. Section 5 summarizes the obtained results. The work is concluded in section 6. A video introducing the problem statement and the proposed approach is available in [Kro13].

2 FEATURE UNCERTAINTY EXTRACTION

In this section we propose a method to extract feature uncertainties from spherical images. This was achieved by decomposing a given spherical image into virtual perspective images. The positions of the appendant virtual cameras were set to the center of the spherical camera and their viewing directions were uniformly distributed throughout the sphere (Figure 2).

In the next step, features were extracted within the resulting perspective images and transformed to latitude-longitude representation of the spherical coordinate system. The performed calculations are hereby equivalent to transformations between world coordinate system and perspective camera coordinate system as outlined in Hartley et al. [HZ00], assuming the spherical camera aligned with world coordinates. The transformation between Cartesian and spherical coordinates is given in [KK00].

Within the perspective images, the uncertainty distribution of the extracted features was obtained based on the

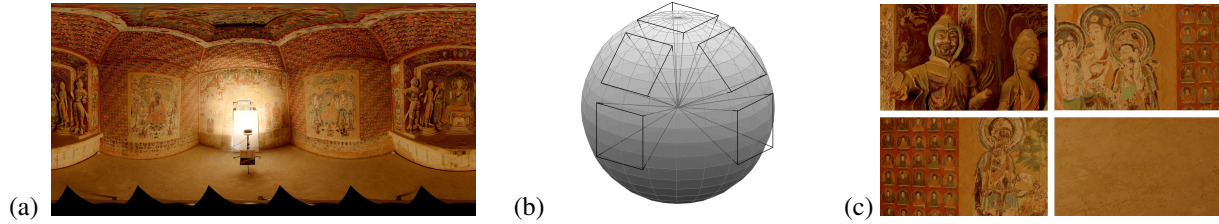


Figure 2: Spherical image ($14'000 \times 7'000$ pixel) of the Mogao cave number 322 in China. A set of 11 spherical HDR-images was acquired to perform pose uncertainty estimation and 3D reconstruction (a). Illustration of the decomposition of a spherical image into perspective ones (b). Resulting perspective images, in which the features are extracted (c). Note, that those cut-outs have to overlap to assure that all image data of the sphere was considered.

method introduced by Zeisl et al. [ZGS⁺09]. The distributions are represented by a 2D covariance matrix such as

$$\sigma_{2D} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{pmatrix}. \quad (1)$$

When describing these uncertainties in latitude-longitude representation of the spherical image, their essential characteristic of being Gaussian distributed is lost (Figure 1(b)). Approximating the resulting distribution by a Gaussian is inaccurate and other descriptions, such as mixture of Gaussians complicate further propagation of the obtained uncertainties.

A representation, which allows an appropriate description of the extracted uncertainties as Gaussians will therefore be presented in the following section.

3 UNCERTAINTY DESCRIPTION

This section introduces the proposed uncertainty representation of spherical images. Instead of representing a given feature in spherical coordinates through latitude-longitude representation, we propose to use a normal vector \mathbf{n} . This vector, described in the 3D Cartesian camera coordinate system is characterized by its perpendicular alignment to the spheres surface and its unit constraint. The previously obtained 2D uncertainties of the image features as specified in equation 1 can then be expressed by assigning an appropriate uncertainty to the normal vector \mathbf{n} .

The requirement of this method is a globally strictly convex and differentiable shape, the image is represented on. This assures that each point on the shape is uniquely identifiable by the normal emanating from the surface at that position. The considered spherical camera as well as other catadioptric camera models meet this requirement.

The position of an image feature \mathbf{P} on the sphere, available in latitude-longitude representation as (θ_P, ϕ_P) -coordinates, can simply be transformed into the corresponding normal vector description \mathbf{n}_P by

$$\mathbf{n}_P = \begin{pmatrix} \sin(\theta_P) \sin(\phi_P) \\ \sin(\theta_P) \cos(\phi_P) \\ \cos(\theta_P) \end{pmatrix}, \quad (2)$$

whereat the condition $\|\mathbf{n}_P\| = 1$ holds.

Any location uncertainty of the 2D point, such as extracted SIFT-location uncertainty, also has to be transferred from the two-dimensional representation in the local perspective images to the 3D normal vector representation. The 3D uncertainty of the normal vector is initialized as

$$\sigma_{\mathbf{n}} = \begin{pmatrix} \sigma_{2D} & \mathbf{0}_2 \\ \mathbf{0}_2^T & \sigma_{\epsilon} \end{pmatrix} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & 0 \\ \sigma_{yx} & \sigma_{yy} & 0 \\ 0 & 0 & \sigma_{\epsilon} \end{pmatrix} \quad (3)$$

by considering equation 1 for the feature uncertainty in 2D. This equation adds a third dimension to the uncertainty representation. When it comes to the implementation of this equation, it is necessary to ensure numerical stability by choosing $\sigma_{\mathbf{n}}$ to be positive-definite through setting $\sigma_{n_{33}} = \sigma_{\epsilon} > 0$. The value of σ_{ϵ} was chosen to

$$\sigma_{\epsilon} \ll \min(\lambda_1, \lambda_2), \quad (4)$$

where λ_1 and λ_2 represent the eigenvalues of σ_{2D} . Figure 3(a) shows a geometrical visualization of $\sigma_{\mathbf{n}}$.

The following calculations outline the correct alignment of the uncertainty distribution $\sigma_{\mathbf{n}}$ with the accordant \mathbf{n} -vector, splitting up into two consecutive steps. Firstly, $\sigma_{\mathbf{n}}$ is converted to comply with the location of the image feature on the sphere as illustrated in figure 3(b). Afterwards, the correct orientation of the uncertainty is ensured as shown in figure 3(c).

The first step is described through the rotation vector \mathbf{r}_A , conforming to the constraint

$$\mathbf{n}_P = R(\mathbf{r}_A) \hat{\mathbf{e}}_z. \quad (5)$$

$R(\cdot)$ represents hereby a rotation matrix based on axis angle by applying the Rodrigues formula [ZF92]. The vector \mathbf{r}_A is obtained through

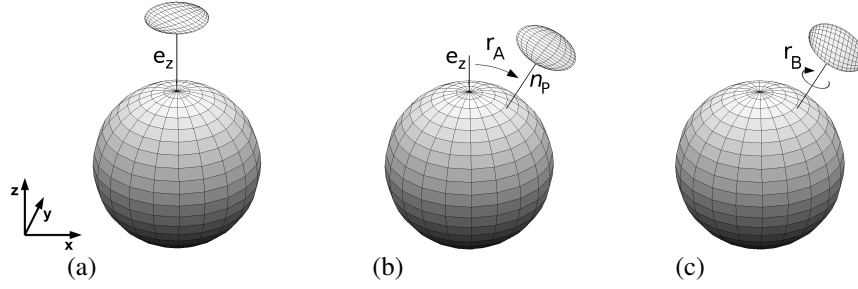


Figure 3: Uncertainty representation with the proposed normal-vector method: An ellipsoid representing the 1σ (68% confidence) neighbourhood of the extracted feature uncertainty is by default initialized at the image pole along the \hat{e}_z -axis (a). Then the uncertainty distribution is aligned with its corresponding location by applying \mathbf{r}_A (b). Finally, the rotation \mathbf{r}_B is applied to ensure the orientation of the originally extracted uncertainties (c).

$$\mathbf{r}_A = \cos^{-1}(\hat{\mathbf{e}}_z \cdot \mathbf{n}_P) \frac{\hat{\mathbf{e}}_z \times \mathbf{n}_P}{\|\hat{\mathbf{e}}_z \times \mathbf{n}_P\|}. \quad (6)$$

The geometrical alignment of \mathbf{r}_A is visualized in figure 3(b) by the dashed line, oriented perpendicular to \hat{e}_z and \mathbf{n}_P . The term $\frac{1}{\|\hat{\mathbf{e}}_z \times \mathbf{n}_P\|}$ normalizes the vector $\hat{\mathbf{e}}_z \times \mathbf{n}_P$ and $\cos^{-1}(\hat{\mathbf{e}}_z \cdot \mathbf{n}_P)$ assigns the length to the vector, which corresponds to the rotation angle. Finally, a further rotation with \mathbf{n}_P as rotation axis is applied through

$$\mathbf{r}_B = \cos^{-1} \left(\frac{\text{sgn}(\mathbf{n}_{P_y}) \cdot \mathbf{n}_{P_x}}{\|\mathbf{n}_{P_{x,y}}\|} \right) \cdot \mathbf{n}_P. \quad (7)$$

Based on this rotation, it is assured that location uncertainties of features are oriented in the way they were extracted from the perspective image (Figure 3(c)). \mathbf{n}_{P_x} determines the x-element of the vector \mathbf{n}_P . $\mathbf{n}_{P_{x,y}}$ represents the vector $(\mathbf{n}_{P_x} \ \mathbf{n}_{P_y})^\top$. The term $\text{sgn}(\mathbf{n}_{P_y})$ identifies the correct quadrant for the evaluation of $\cos^{-1}(\cdot)$.

4 APPLICATION: ESTIMATION OF CAMERA POSE UNCERTAINTY

To evaluate the capability of our approach, we estimated the pose uncertainty of a spherical camera based on our proposed representation and compared the results against those obtained with latitude-longitude representation. The camera pose uncertainty is calculated based on correspondences between features of the spherical image and 3D points in the world coordinate system (wcs). Without loss of generality, we choose the wcs and the camera coordinate system (ccs) of the spherical camera to be congruent. This simplifies the transition of a 3D point \mathbf{M}^w in wcs to \mathbf{M}^c in ccs given through

$$\mathbf{M}^c = \mathbf{R}^{cw} \mathbf{M}^w + \mathbf{t}^{cw} \quad (8)$$

to an identity operation $\mathbf{M}^c = \mathbf{M}^w$, whereas the superscript $[\cdot]^{cw}$ identifies the direction of transformation

from wcs to ccs. Additionally \mathbf{R} and \mathbf{t} identify the rotation and translation of wcs against ccs.

The image points \mathbf{M}^s of the spherical image are finally obtained by projecting all points \mathbf{M}^c through the central projection point of the camera, given as \mathbf{t}^{cw} , onto a unit sphere. Based on this step, 2D-3D correspondences between all points \mathbf{M}^s in 2D latitude-longitude representation and \mathbf{M}^w in wcs are established.

Our evaluations have been performed on synthetical and real data. For synthetical evaluations, n 3D points in wcs as well as the camera model and the image features were generated with Matlab[®]. For evaluation with real data, spherical images of calibrated cameras were chosen: The camera pose had previously been calibrated using an SfM-approach for spherical images and was optimized by minimizing the squared reprojection error between image points and 3D points, projected into the image. Software such as the *Sparse Bundle Adjustment Package* provided by Lourakis and Argyros [LA09] perform this task efficiently. For the minimization of the reprojection error, the measurement function h was chosen according to Paganì and Stricker [PS11] as

$$\sum_{j=1}^n h(\mathbf{M}^{s_j}, \mathbf{M}^{c_j}) = \sum_{j=1}^n \cos^{-1} \left(\frac{(\mathbf{M}^{s_j})^T \mathbf{M}^{c_j}}{\|\mathbf{M}^{c_j}\|} \right), \quad (9)$$

which corresponds to the geodesic distance between the 2D-3D correspondences. The point \mathbf{M}^{s_j} denotes hereby the j th image point on the spherical image, represented as 3D unit vector, \mathbf{M}^{c_j} identifies the j th 3D point in ccs. $\left(\frac{(\mathbf{M}^{s_j})^T \mathbf{M}^{c_j}}{\|\mathbf{M}^{c_j}\|} \right)$ represents finally the normalized scalar product between \mathbf{M}^{s_j} and \mathbf{M}^{c_j} .

For the calculation of the pose uncertainties, we assume an optimization to be performed based on equation 9. Since such an optimization does not deliver perfect agreement of image features and backprojected

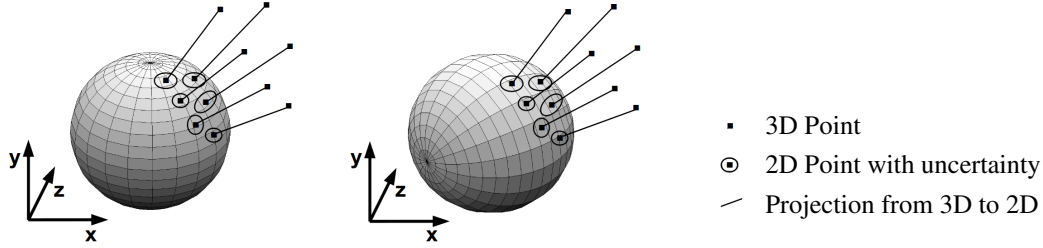


Figure 4: Two omnidirectional cameras in identical environments. 3D points are projected into the spherical images and uncertainties are attached to them. Based on their location on the sphere, their accuracy of description varies when using latitude-longitude representation, depending on their latitude-longitude position.

3D points for real data, this fact was also considered in our simulations. Image features were firstly generated through backprojection of 3D points. After applying Gaussian noise to those 3D points, they were once again backprojected creating deviation between backprojected 3D points and image features.

The uncertainty of the camera pose for simulated and real data was then obtained by performing a first order error propagation with respect to the covariance matrices of \mathbf{t}^{cw} and \mathbf{R}^{cw} . Without loss of generality, in the current work evaluations are limited to the translation covariance of the camera to proof the reliability of the proposed representations.

The derivation of the needed Jacobian matrices from the measurement function h for performing camera pose uncertainty estimation is outlined below and implies the following preconditions: Uncertainties of all error-prone measurements, such as image points \mathbf{M}^s on the sphere and 3D world points \mathbf{M}^w are assumed to be Gaussian distributed. The subsequent uncertainty propagation is furthermore restricted to linear propagation methods. It is assumed, that the camera pose was optimized towards its environment by minimizing the reprojection error of 2D3D-correspondences between image points \mathbf{M}^{s_j} and world points \mathbf{M}^{w_j} , which have to be transferred to the camera coordinate system (ccs) by applying equation 8. This optimization step can reliably be performed by software-packages based on the Levenberg-Marquardt-algorithm, such as SBA [LA09]. For quantifying the reprojection error between the 2D3D correspondences, the geodesic error as measurement function h is used (Equation 9).

After the optimization step, the uncertainty of the camera pose σ_{pose} is calculated using the retrieved results as linearization point. Bleser et al. [BBS07] present those calculations for perspective cameras. Since we limit our calculations to the translational component of the pose uncertainty, we obtain for spherical cameras and n 2D3D-correspondences

$$\sigma_t \approx \left(\sum_{j=1}^n \left(\frac{\partial h}{\partial \mathbf{t}_j^{cw}} \right)^\top P_j^{-1} \frac{\partial h}{\partial \mathbf{t}_j^{cw}} \right)^{-1}. \quad (10)$$

For $\partial h / \partial \mathbf{t}^{cw_j}$ we obtain

$$\frac{\partial h}{\partial \mathbf{t}^{cw_j}} = \frac{1}{\xi_j} \left(\frac{1}{\kappa_j} \mathbf{M}^{c_j} \left((\mathbf{M}^{s_j})^\top \mathbf{M}^{w_j} \right) - \frac{\mathbf{M}^{s_j}}{\|\mathbf{M}^{w_j}\|} \right), \quad (11)$$

with

$$\kappa_j = \left((\mathbf{M}_x^{w_j})^2 + (\mathbf{M}_y^{w_j})^2 + (\mathbf{M}_z^{w_j})^2 \right)^{3/2} \quad (12)$$

and

$$\xi_j = \sqrt{1 - \left(\frac{(\mathbf{M}^{s_j})^\top \mathbf{M}^{w_j}}{\|\mathbf{M}^{w_j}\|} \right)^2} \quad (13)$$

since $\frac{\partial \cos^{-1}(\alpha)}{\partial \alpha} = -\frac{1}{\sqrt{1-\alpha^2}}$.

P_j is approximated as

$$P_j \approx \begin{pmatrix} \frac{\partial h}{\partial \mathbf{M}^{w_j}} & \frac{\partial h}{\partial \mathbf{M}^{s_j}} \end{pmatrix} \begin{pmatrix} \sigma_{\mathbf{M}^{w_j}} & 0 \\ 0 & \sigma_{\mathbf{M}^{s_j}} \end{pmatrix} \begin{pmatrix} \frac{\partial h}{\partial \mathbf{M}^{w_j}} \\ \frac{\partial h}{\partial \mathbf{M}^{s_j}} \end{pmatrix}. \quad (14)$$

$\sigma_{\mathbf{M}^{w_j}}$ and $\sigma_{\mathbf{M}^{s_j}}$ describe the uncertainties of the world and image points as Gaussians. The matrix $\partial h / \partial \mathbf{M}^{w_j}$ represents the Jacobian of h with respect to \mathbf{M}^{w_j} and $\partial h / \partial \mathbf{M}^{s_j}$ the equivalent with respect to \mathbf{M}^{s_j} :

$$\frac{\partial h}{\partial \mathbf{M}^{w_j}} = \frac{1}{\xi_j \kappa_j} \left((\mathbf{M}^{s_j})^\top \mathbf{M}^{w_j} \right) \left((\mathbf{R}^{cw})^\top \mathbf{M}^{w_j} \right) - \frac{1}{\xi_j} \frac{(\mathbf{R}^{cw})^\top \mathbf{M}^{s_j}}{\|\mathbf{M}^{w_j}\|} \quad (15)$$

$$\frac{\partial h}{\partial \mathbf{M}^{s_j}} = -\xi_j \|\mathbf{M}^{w_j}\| \cdot \mathbf{M}^{c_j} \quad (16)$$

Based on this, the resulting pose uncertainties of the spherical cameras were evaluated for the common latitude-longitude representation as well as for the normal-vector representation. The results for the calculated pose uncertainties will be detailed in the next section.

5 EVALUATION AND RESULTS

This section presents results of pose uncertainty estimation for spherical cameras in several scenarios.

Since omnidirectional cameras are able to register image features in arbitrary directions, the calculated pose uncertainty can be expected to be independent from the camera orientation \mathbf{R}^{cw} , when considering \mathbf{t}^{cw} to be constant. This is not the case, when choosing latitude-longitude representation for the uncertainty calculation (Figure 4).

The reason for this is illustrated in figure 1(b), exemplifying that a Gaussian distribution on a sphere cannot be adequately described by a Gaussian near the image poles. This leads to the conclusion, that the (θ, ϕ) -representation is not valid to reliably approximate error distributions with Gaussians.

To highlight this, we reduced the distribution of artificially generated image features in our evaluations with synthetic data to a very limited solid angle of the spherical image, emphasizing the influence of changing camera orientation for (θ, ϕ) -representation. The parameter σ_ϵ specified in equation 4 was chosen as $\sigma_\epsilon = 10^{-10} \cdot \min(\lambda_1, \lambda_2)$ throughout the evaluations. This choice of σ_ϵ is justified by evaluations showing negligible impact to the results with $\sigma_\epsilon < 10^{-8}$.

Figure 5(a-e) visualizes the resulting pose uncertainty for rotation around the \hat{e}_x -axis and the \hat{e}_z -axis for synthetic data: The evaluation was based on a distribution of $n = 20$ 3D points \mathbf{M}^v . The coordinates of those points as well as the vector \mathbf{t}^{cw} were kept constant in wcs throughout the evaluation process. Only the orientation of the omnidirectional camera was rotated in steps of 10° by varying \mathbf{R}^{cw} accordingly. With each iteration step, the distance of the points towards the pole changed.

Due to the distortion in latitude-longitude representation, the image feature uncertainties enter differently into the pose calculations. Results of pose uncertainty calculation at the image poles differ up to 27% from the uncertainty obtained at the image equator. Reason for this disparity is an overestimation of uncertainties at the poles when using latitude-longitude-representation. The resulting twofold symmetry of the pose uncertainty is explained by the fact, that the point distribution passes both poles during the evaluation steps. Throughout the whole evaluation process, the pose uncertainty calculated based on the normal vector representation is

unchanged and complies with the expectation of obtaining constant results.

Our evaluation with real data was based on two datasets: The first dataset contains a set of 10 images of the Mogao cave number 322 in China. The second dataset consists of 183 images taken of a street crossing in Berlin (Oberwallstraße, Jägerstraße). All images were taken as high-dynamic range (HDR) images and have a resolution of $14'000 \times 7'000$ pixel. Therefore up to 108'000 image features were extracted in a single image and considered for the calculation of the camera pose uncertainty.

Since the extracted features were distributed uniformly throughout the whole image, the camera pose uncertainty in conventional latitude-longitude representation varied less compared to the synthetic scenario. When using standard low-dynamic range images, a non-uniform distribution of features throughout the images is more likely, leading to higher disparities between the presented approaches. Figure 5(f) shows a typical result for a spherical image.

6 CONCLUSION AND FUTURE WORK

A method for uncertainty representation of spherical image features based on normal vectors has been proposed and compared to the commonly used latitude-longitude representation. It was shown, that this method - in contrast to latitude-longitude representation - is able to reliably represent image features throughout the whole sphere, since the used normal vector concept is not subject to distortions, discontinuities or nonlinear effects at the image poles.

Basing the pose uncertainty estimation of spherical cameras on the proposed approach improves its reliability and makes the quality of its results comparable to perspective estimates, such as [BBS07].

The proposed method is furthermore expected to be robust against up and down-sampling of the images, as long as a consistent uncertainty extraction of the image features is assured for different scales (e.g. by using scale-invariant SIFT [Low04], SURF [BTVG06] or ORB [RRKB11] features). A detailed evaluation has to be considered as future work.

Also the use of the proposed uncertainty representation, e.g. as additional information for surface fitting, methods for uncertainty propagation through subsequent reconstruction algorithms such as sparse reconstruction and pointcloud generation will be part of future work.

7 ACKNOWLEDGEMENTS

This work was partially funded by the BMBF-projects CAPTURE (01IW09001) and DENSITY (01IW12001).

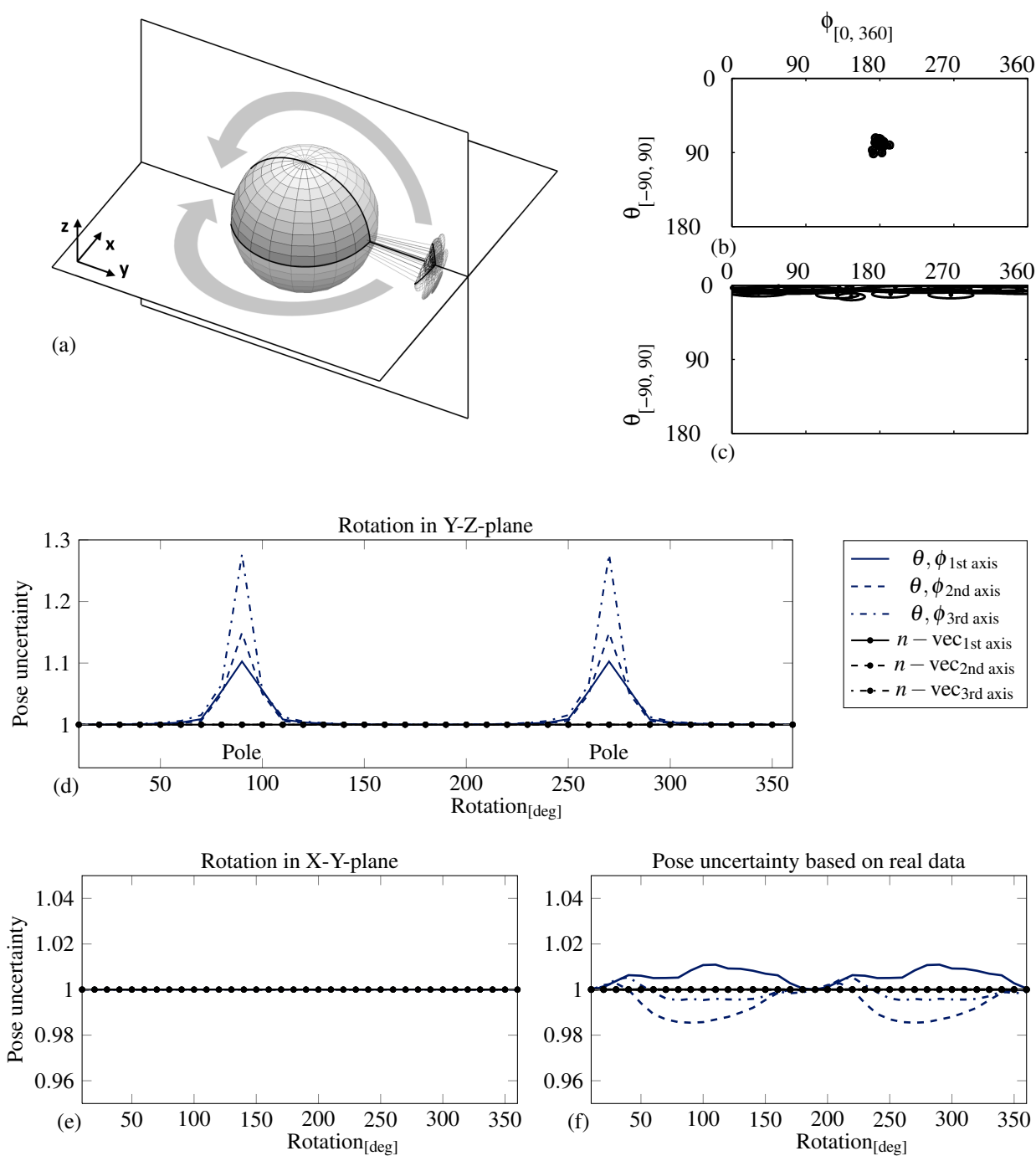


Figure 5: Subfigure (a) shows a synthetic point distribution M^s represented as normal vectors and two exemplarily evaluated rotation planes, the vectors were rotated in. Point uncertainties in latitude-longitude representation are visualized, when being located at the equator (b) and at the image poles (c). Subfigure (d) and (e) show the resulting camera pose uncertainties, expressed through the eigenvalues of the camera pose uncertainty matrices. These eigenvalues furthermore represent the expansion of the uncertainty ellipsoids of the cameras along the three principle axis (1st, 2nd, 3rd axis) and were normalized to the first evaluation step. Subfigure (f) finally illustrates a typical pose uncertainty distribution for a real spherical image with 101778 extracted features. Note that within the figures (d)-(f) all normal vector graphs (*abbrev.: n-vec*) overlap in a single plotted line.

A REFERENCES

- [ADF⁺10] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver. Google street view: capturing the world at street level. *IEEE Computer*, 43(6):32–38, 2010.
- [AY11] M. Ambai and Y. Yoshida. Card: Compact and real-time descriptors. *ICCV*, 2011.
- [BBS07] Gabriele Bleser, Mario Becker, and Didier Stricker. Real-time vision-based tracking and reconstruction. *Journal of Real-Time Image Processing*, 2:161–175, 2007.
- [BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *ECCV*, 2006.
- [CF05] Eduardo Bayro Corrochano and Wolfgang Förstner. Uncertainty and projective geometry. In *Handbook of Geometric Computing*, pages 493–534. Springer Berlin Heidelberg, 2005.
- [CZZF97] G. Csurka, C. Zeller, Z. Zhang, and O.D. Faugeras. Characterizing the uncertainty of the fundamental matrix. *Computer Vision and Image Understanding*, 68(1):18–36, 1997.
- [DLLP11] G. Di Leo, C. Liguori, and A. Paolillo. Covariance propagation for the uncertainty estimation in stereo vision. *Instrumentation and Measurement, IEEE Transactions on*, 60(5):1664–1673, 2011.
- [Gad10] K Gade. A non-singular horizontal position representation. *Cambridge Univ Press*, 2010.
- [GRMG11] D. Gutierrez, A. Rituerto, JMM Montiel, and JJ Guerrero. Adapting a real-time monocular visual slam from conventional to omnidirectional cameras. *OMNIVIS*, 2011.
- [Har98] R.M. Haralick. Propagating covariance in computer vision. *Performance Characterization in Computer Vision*, 1:95–115, 1998.
- [HZ00] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.
- [JU04] S.J. Julier and J.K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.
- [KK00] G.A. Korn and T.M. Korn. *Mathematical handbook for scientists and engineers: definitions, theorems, and formulas for reference and review*. Dover Pubns, 2000.
- [Kro13] B. Krolla. http://www.youtube.com/watch?v=0oOr0V4u_e4, 2013.
- [LA09] M.I. A. Lourakis and A.A. Argyros. Sba: A software package for generic sparse bundle adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009.
- [LCS11] S. Leutenegger, M. Chli, and R.Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. *ICCV*, 2011.
- [Low04] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [PS11] A. Pagani and D. Stricker. Structure from motion using full spherical panoramic cameras. *OMNIVIS2011*, 2011.
- [RRKB11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. *ICCV*, 2011.
- [SK05] B. Streckel and R. Koch. Lens model selection for visual tracking. *Pattern Recognition*, pages 41–48, 2005.
- [Stu64] J. Stuelpnagel. On the parametrization of the three-dimensional rotation group. *SIAM review*, 6(4):422–430, 1964.
- [TIO05] A. Torii, A. Imiya, and N. Ohnishi. Two- and three-view geometry for spherical cameras. In *Proc. Workshop Omnidirect. Vis*, pages 1–8. Citeseer, 2005.
- [TLF10] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(5):815–830, 2010.
- [VR] Spheron VR. SceneCam[®] HDR <http://www.spheron.com/?id=107>.
- [Wei] AG Weiss. Civetta camera <http://www.weiss-ag.org/business-divisions/360-technology>.
- [ZF92] Zhengyou Zhang and Olivier Faugeras. *Three-D-Dynamic Scene Analysis: A Stereo Based Approach*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1992.
- [ZGS⁺09] B. Zeisl, P. Georgel, F. Schweiger, E. Steinbach, and N. Navab. Estimation of location uncertainty for scale invariant feature points. *BMVC*, 2009.

Adaptive Surface Reconstruction for SPH using 3-Level Uniform Grids

Gizem Akinci Nadir Akinci Edgar Oswald Matthias Teschner
 gakinci,nakinci,oswald,teschner@informatik.uni-freiburg.de
 University of Freiburg
 Georges Koehler Allee 052
 79110 Freiburg Germany

ABSTRACT

The marching cubes algorithm is a popular method for constructing surfaces from SPH data sets. In order to preserve all of the surface details in high curvature regions and to prevent potential temporal coherence artifacts, the resolution of the underlying uniform MC grid should be set up sufficiently high. However, this requirement unnecessarily increases the resolution in relatively flat regions where the surface can be constructed with lower resolutions without changing the quality. Accordingly, excessive number of triangles are generated, the memory consumption increases dramatically, and the performance decreases. In this paper, we present a 3-level grid structure which adapts its cells according to the curvature of the fluid surface. In contrast to widely-used octrees, we propose a simple to construct yet efficient hierarchical uniform grid structure. Mesh blocks from different resolution cells are seamlessly stitched by closing cracks with new triangles which establish only 0.15% to 0.6% of overall number of triangles in average. Experiments show that in contrast to the single level low resolution uniform grid approach, the presented method reconstructs fine details properly with a comparable performance; while it produces similar results with less number of triangles, up to four times better memory consumption and up to 60% better performance when compared to the single level high resolution uniform grid approach.

Keywords

particle-based fluids, surface reconstruction, marching cubes, multi-level uniform grids, surface curvature, cracks

1 INTRODUCTION

Generating triangle meshes using the marching cubes algorithm (MC) [LC87] is a common approach for both static point clouds and dynamic particle data. However, the chosen grid resolution restricts the user since surface details are reconstructed properly only by using high resolution grids at the expense of performance, memory footprint and storage. This issue has prompted many researchers to investigate adaptive mesh refinement techniques, e.g. by using octrees [WvG92, SFYC96, WKE99, VT01, LY04, JU06, Man10]. Although being efficient in terms of memory consumption and storage reduction, the construction of adaptive structures is not straightforward, and they lead to cracks in between different resolution cells which need to be handled carefully. The use of uniform grids, on the contrary, is motivated by the simplicity of the structure which is advantageous for especially

dynamic scenes since it allows fast rebuilding of the data structure. To the best of our knowledge, most of the presented adaptive approaches focus on static scenes, e.g. medical visualizations or CAD models, and there are only few researchers who aim to efficiently rebuild those data structures for dynamic scenes, e.g. [ZGHG11].

Our contribution. In this paper, we present a memory efficient and performance friendly multi-level uniform grid structure for Smoothed Particle Hydrodynamics (SPH) surface reconstructions which allows for fast rebuild in dynamic scenes.

In our technique, the particle data set is covered by an axis aligned bounding box (AABB) which is initially subdivided uniformly with coarse (level-1) cells. These cells are utilized to extract the narrow-band region where the surface is actually defined. Depending on the surface curvature, coarse surface cells are subdivided by one (level-2) or two more levels (level-3). This allows for preserving the surface details even on highly turbulent, high curvature parts by using less triangles, since relatively flat parts are treated as level-2 parts. According to our experiments, the explained data structure is sufficient for a proper surface reconstruction of any fluid data set; and the fourth level is not necessary since it does not improve the quality further but

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

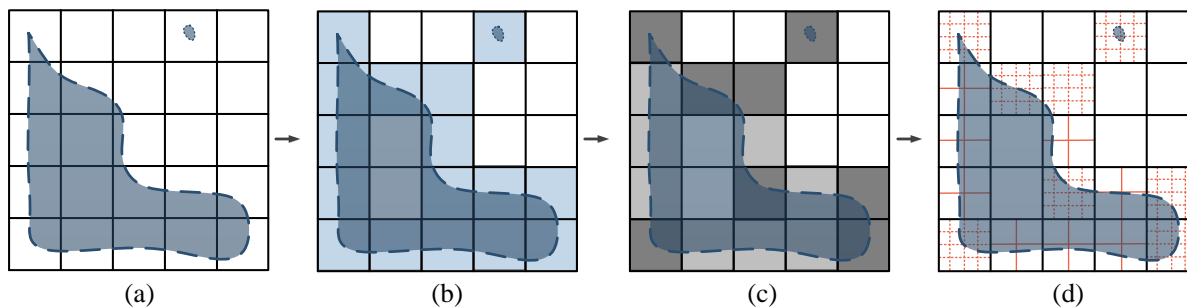


Figure 1: The 3-level adaptive grid. (a) The bounding box of the fluid is used to generate the coarse level uniform grid (level-1). (b) Cells that contain the fluid surface (dashed blue line) are extracted. These surface cells are shown in blue color. (c) Surface cells with low surface curvature are marked as level-2 (light gray cells) and the other surface cells are marked as level-3 (dark gray cells). Cells around the isolated pieces are always considered as level-3. (d) Level-2 cells are subdivided by one more level (straight red lines) while level-3 cells are subdivided by two more levels (dashed red lines).

reduces the performance significantly. The described three levels are illustrated in Fig. 1. We close the arisen cracks eventually by creating new triangles in between different resolution mesh blocks.

Experiments show that when compared to high resolution single level uniform grids, the presented method produces similar results with up to four times better memory consumption and up to 60% better computation time. Two different test scenarios are discussed in Sec. 4, where the computation time and memory consumption data are given for all scenes.

2 RELATED WORK

There exist various approaches that address the visualization of surfaces for fluids or unorganized point data sets that can also be applied to particle based fluids, e.g. implicit surface tracking [WH94], explicit surface tracking [Mul09, BB09], surface splatting [ZPvBG01, ALD06, vdLGS09], screen space meshes [MSD07], rendering using raycasting [MSD07, FAW10, GSSP10] or surface generation using voronoi diagrams [RS09]. Our approach contributes to the field of generating enclosed triangulated fluid surfaces using the marching cubes approach with an adaptive mesh generation technique.

One way to generate an adaptive mesh is to use octree structure which was addressed by many researchers, e.g. [WvG92, SFYC96, WKE99, VT01, LY04, JU06, Man10]. While being very efficient in terms of memory consumption and storage reduction, the construction of octrees is time consuming. Dynamic update of octrees, e.g. shrinking or enlarging the grid and updating child cells, in particular can be time consuming, as one can probably end up with total rebuild of the octree; which makes them less feasible for dynamic scenes. However, we aim a data structure which allows for a fast total rebuild. Furthermore, random access to octree cells usually take logarithmic time, which

causes additional overhead. In addition, octrees produce many different resolution cells which means that two leaf cells may differ more than one level. In such a case, crack handling gets difficult, which is not an issue for our data structure. A similar discussion can be found in [Bri03] where Bridson proposes an alternative grid-based method that focuses on the narrow-band region by using only one level of detail.

As mentioned earlier, if levels of two neighboring cells differ, cracks arise in the corresponding transition faces. There exist different approaches which address this problem. Using simple crack patching [SFYC96, VKSM04], points that reside on the high resolution edge of one cell are projected on the low resolution edge of the neighboring cell. However, this technique produces T-vertices which may lead to visual artifacts during rendering. Filling cracks with new triangles is another popular method for handling cracks. Westermann et al. [WKE99] proposed a method where cracks are fixed by replacing coarse triangles by fans of triangles. Ju and Udeshi [JU06] prevented cracks by adding new polygons using a hybrid method that uses the marching cubes and dual contouring. It is stated in [JU06] that the mesh size can get too large by using this approach after newly added triangles. Later, Lengyel [Len10] presented transition cells method where new cells are added in between two different resolution cells for generating new triangles. The newly generated transition cells have to be checked for many cases which is a time consuming process.

Although conventional adaptive structures reduce the storage requirements efficiently, more effective results can be obtained by incorporating hashing. However, this is a challenging task and only few approaches have been proposed so far which address either hashing of octrees [Sigg06] or hashing of multi-level grids for raytracing, e.g. [CPJ10, LD08]. We also get assistance from hash maps for keeping our newly generated fine resolution grid points in each coarse cell (see Sec. 3.2),

while we leave our top level coarse grid un-hashed for a straightforward implementation.

Memory footprint can be reduced by also applying dynamic tubular grids (DT-grid) [NM04] or run-length encoding (RLE) [HWB04] schemes, which are rather challenging to implement in comparison to alternative grid- and tree-based data structures.

Parallelization of the surface reconstruction algorithm is another topic which has been gaining attention in recent years. Zhou et al. [ZGHG11] proposed a parallelization technique for octrees which runs entirely on GPU. Later, Akinci et al. [AIAT12] presented a parallel method which reconstructs the surface of particle-based fluids in the narrow-band region and runs either on CPU or GPU. Memory footprint is still an open issue to be improved in this method since it sticks to single level uniform grids. While being very efficient in terms of performance, the incorporation of hashing in both approaches is challenging due to the potential thread safety issues.

The quality of the reconstructed surface depends not only on the resolution of the underlying grid but also on the preferred scalar field computation method. Within the context of these techniques, Zhu and Bridson [ZB05] proposed the signed distance field approach which alleviates former bumpiness issues but suffers from artifacts in concave regions. Solenthaler et al. addressed this issue in [SSP07]. In [YT10], Yu and Turk presented an anisotropic kernel approach which results in high quality, less bumpy surfaces while being computationally expensive. Recently, Bhattacharya et al. [BGB11] proposed a surface reconstruction technique which is based on a level set method. This method outputs surface approximations and performs smoothing steps that finally generate rough surfaces which causes to lose the details of the input particle set. With these issues in mind and based on the comparisons given in [AIAT12] and [AAIT12], we decided to use the method of Solenthaler et al. [SSP07] in all of our test scenes.

To the best of our knowledge, the efficient implementation of adaptive structures for particle-based fluids is limited with the work of Zhou et al. [ZGHG11] which uses octrees unlike our method. In this paper we present a method which allows for fast rebuild of the grid and is suitable for dynamic scenes in terms of both memory efficiency and performance.

3 SURFACE RECONSTRUCTION USING 3-LEVEL GRIDS

In this section, we present an adaptive surface reconstruction method for SPH using 3-level uniform grid.

According to our experiments, in order to obtain proper surface reconstructions, MC grid cell size should not

be larger than $2r$, with r being the radius of the fluid particles, i.e. the half of the particles' equilibrium distance. However, even $2r$ cell size can be insufficient to preserve all of the surface details. In such a case, experiments show that the cell size of r preserves all surface features appropriately. However, using such a small cell size throughout the whole scene causes a trade-off between the surface quality and the performance-memory consumption. Besides, it is clear that on relatively flat regions, the cell size of $2r$ is already sufficient for obtaining proper results. Therefore, we reconstruct the fluid surfaces using these two resolutions in different regions depending on the surface details.

We initialize our surface reconstruction steps by extracting the narrow-band region where the surface is actually defined. Performing this operation does not require a high resolution grid. We initially create our grid using the cell size of $4r$, extract the narrow-band region using the coarse cells in this resolution, and subdivide the found surface cells using the surface curvature information.

We extract the surface cells using the method of Akinci et al. [AIAT12]. We refer the reader to [AIAT12] for details, however, we briefly outline the steps here. According to this method, any grid point that is in the proximity of a surface particle is defined as a surface point. Therefore, we traverse through all surface particles, define an AABB which spans $4r$ length in x , y , z directions around each surface particle and mark all grid points that reside in this AABB as surface points. Each surface point is used to mark the corresponding cell whose lower left corner is initiated in this point as a surface cell. Different from [AIAT12], we mark the cells as splash cells instead of surface cells if they are in the neighborhood of splash particles, i.e. particles whose number of neighbors are less than a pre-defined value which is 3 in our test scenes. By doing this, we are easily able to exclude such cells from the curvature computation in future steps and gain performance since they are directly subdivided as level-3 high resolution cells.

Surface particle extraction is performed in the preprocessing step using the smoothed color field method of Muller et al. [MCG03]. The same method is also used to compute particles' normals which are required for the curvature computation in the following step (see Sec. 3.1). After refining the cells, the scalar field is computed over all grid cells (see Sec. 3.2), the cracks are handled by adding new triangles and different resolution meshes are stitched seamlessly (see Sec. 3.3).

3.1 Curvature Computation

According to our criterion, a coarse surface cell can be subdivided as either level-2 or level-3 which depends on the surface curvature inside the cell. We approximate the surface curvature using the method described

in [IAAT12]. Hence, we firstly compute the curvature for each surface particle that resides in this cell with the help of its neighboring particles as:

$$\kappa_i = \sum_j \kappa_{ij} = \sum_j (1 - \hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j) W(\mathbf{x}_{ij}, h) \quad (1)$$

where j stands for the neighboring particles, $\hat{\mathbf{n}}$ is the normal of any particle and W is the kernel function described as:

$$W(\mathbf{x}_{ij}, h) = \begin{cases} 1 - \|\mathbf{x}_{ij}\|/h & \text{if } \|\mathbf{x}_{ij}\| \leq h \\ 0 & \text{else} \end{cases} \quad (2)$$

with $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$.

Finally, the curvature of any surface cell is approximated as:

$$\kappa_{cell} = \left(\sum_{\text{for all } i} \kappa_i \right) / N \quad (3)$$

where i and N represent the surface particles and the total number of surface particles inside the cell, respectively.

If the curvature is smaller than the predefined threshold, we mark the cell as low resolution level-2 cell. Otherwise, the cell is marked as high resolution level-3 cell. In the remainder of the paper, we will call any coarse surface cell as level-2 or level-3 cell depending on its refinement level.

3.2 Scalar Field Computation

We initialize the second step of our method with cell refinement. Therefore, we traverse through all coarse surface cells. If the cell is marked as a level-2 cell, we generate 8 new cells in the coarse cell which correspond to 27 new grid points. A similar approach is followed for level-3 cells with 64 new cells and 125 new grid points. A 2-dimensional illustration of this refinement is shown in Fig. 2.

At this part of our implementation, we need a data structure to keep our newly generated grid points. This structure should support easy access to the required grid point. For this aim, we compute an id for each grid point and use hash map structure which allows for easy data access by querying the ids that are the keys of the hash map. This structure also stores the corresponding scalar values as the data part of each entry. The id of a grid point is computed using its position $\mathbf{GridPos}_{l-3} = (gpx, gpy, gpz)$ and number of potential grid points X_{l-3} and XY_{l-3} in x - and xy -directions, respectively, in a virtual, high resolution level-3 grid throughout the whole scene. $\mathbf{GridPos}_{l-3}$ can be written as:

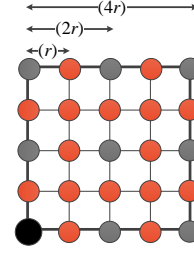


Figure 2: A 2-dimensional illustration of the cell refinement. The thick black line shows the outline of the coarse surface cell whose initial coarse point is colored in black. The gray circles demonstrate level-2 points and both gray and red circles demonstrate level-3 points that are added after the cell refinement.

$$\mathbf{GridPos}_{l-3} = 4 \cdot \mathbf{vc} + \mathbf{offset} \quad (4)$$

where \mathbf{vc} is the position of the coarse initial grid point of the cell in the coarse grid. The 3-dimensional $\mathbf{offset} = (a, b, c)$ vector of any new grid point describes the position of this point in the cell. a , b and c values are defined within the range of $[0,5)$. Thus, these values start with 0 and depending on the refinement level, they increase by 1 or 2 for the high resolution and low resolution cells, respectively (see Fig. 3-(a)). Furthermore, X_{l-3} and XY_{l-3} are computed as:

$$\begin{aligned} X_{l-3} &= 4 \cdot ncx + 1 \\ Y_{l-3} &= 4 \cdot ncy + 1 \\ XY_{l-3} &= X_{l-3} \cdot Y_{l-3} \end{aligned} \quad (5)$$

with ncx and ncy being the number of cells in x and y directions in our coarse grid. Finally, the id of a grid point is computed as:

$$id = gpx + X_{l-3} \cdot gpy + XY_{l-3} \cdot gpz \quad (6)$$

After computing the id of any grid point, we check whether it is already stored in the hash map. In such a case, we do not compute the scalar value since this information is already kept together with the id . Otherwise, we compute the scalar value of the grid point using the improved signed distance field approach of Solenthaler et al. [SSP07]. At this step, it is important to identify the particles which contribute to the scalar value computation of each grid point. Particles within the influence radius of each coarse grid point can be easily retrieved by using the method of Akinci et al. [AIAT12]. However, since the influence region is changed for the newly added fine points, we check all the particles that lie also in the influence region of the final grid point in the cell, i.e. the fine point with $\mathbf{offset}(4,4,4)$, which is shown with yellow color in Fig. 3. The computed scalar value can now be stored in our hash map with its corresponding id .

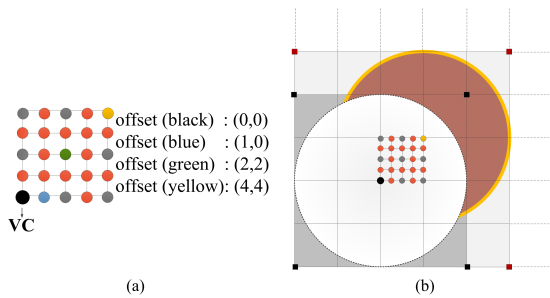


Figure 3: (a) The coarse initial grid point with position vc is illustrated with a black circle. Offset values for some fine grid points are given in 2-D which is similar for the 3-D version. (b) The white, large circle shows the influence region of the coarse initial surface grid point. However, this region is not sufficient for newly added fine grid points. The red region which is the influence region of the yellow point in the figure should also be checked for influencing particles.

3.3 Triangulation

Triangulation is the most challenging part of our method, since we need a special treatment for handling the cracks which arise in between different resolution cells. As mentioned before, various crack handling techniques have been proposed in the literature. Many of these methods, however, are either computationally expensive, or produce T-vertices which cause visual artifacts, or produce large number of triangles and cause memory inefficiency. Therefore, we present a method for an efficient crack closing. We perform this step gradually by ensuring the scalar field continuity first and covering the cracks with new triangles afterwards.

3.3.1 Scalar Field Continuity

The scalar field that is computed over our multi level grid is not continuous, i.e. the scalar field continuity is broken on the faces of neighboring different resolution cells. Therefore, similar to [OR97], we adjust the values of intermediate grid points on such transition cell faces by sub-sampling the original data (see Fig. 4). So as to carry out this task in our implementation, we traverse through all level-2 cells and identify the cells that have a level-3 cell neighbor. If any level-2 cell meets this condition, we adjust the values in the corresponding face.

Even though this approach alleviates the scalar field discontinuity, cracks can still occur due to the numerical sampling problems which are generally observed after the computations around the middle grid point of the face, e.g. the pink grid point in Fig. 4. Those cracks are covered with triangles in our method which is explained in the following section.

3.3.2 Crack Problem

At the final step of our method, we close the cracks with new triangles for a proper visualization.

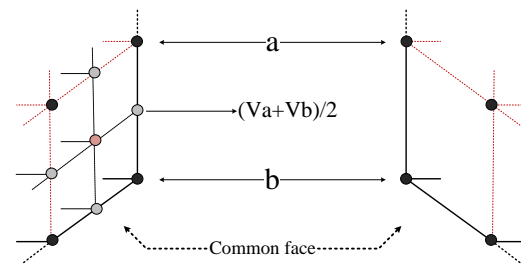


Figure 4: The scalar field continuity is ensured on transition cell faces by sub-sampling the original data. V_a and V_b are the scalar values of grid points a and b , respectively.

In the beginning of our implementation, we apply standard marching cubes algorithm to all level-2 cells. Subsequently, we visit each level-3 cell and handle them sub-cell by sub-cell (see Fig. 5-(a)). We keep the face information of each sub-cell relative to the coarse cell in which it lies. For instance, the sub-cell which is shown as the first sub-cell in Fig. 5-(a), shares its near, left and bottom faces with the coarse cell that hosts it. Then, we check whether any of these shared faces has a level-2 cell neighbor using the cell neighborhood information of the coarse cell. As soon as the condition is met, we determine the edges (see Fig. 5-(b)) which have a potential surface intersection point on itself that will be computed by the marching cubes algorithm. It is sufficient to check the scalar values of the end points for each edge to determine potential intersections. According to our criterion, if there is an intersection on at least one of the inner edges (e8...e11), then there exists a crack at that face (see Fig. 6). Therefore, we keep the intersected edge information of each sub-cell face.

Once all of the required information is gathered for the crack handling, we apply marching cubes algorithm to all eight level-3 fine cells of the sub-cell. Later, we check each face of our sub-cell to see if there is any previously marked intersection on inner edges. In this case, we firstly create a crack array which is necessary to keep the intersected edge ids, and then we perform the following steps: 1) Go through outer edges

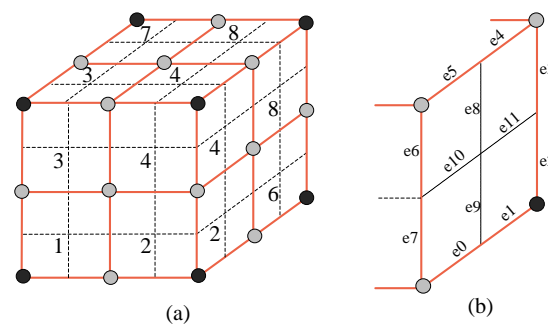


Figure 5: (a) Eight sub-cells of a level-3 cell are illustrated by red cells. (b) Edges are illustrated with their ids for the right transition face of sub-cell-6.

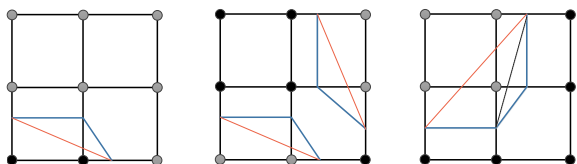


Figure 6: Three of the possible configurations for crack formation. Gray and black colored points have scalar values that are either less or more than the specified isovalue, respectively. Blue lines are generated on high resolution face while red lines are the results of low resolution face. Either one or two cracks can arise (left and middle) or only one crack can cause two triangles (right).

(e0...e7). Push the id of the first found edge which has an intersection point into the crack array. 2) For the found outer edge, check neighboring inner (NI), opposing inner (OI) and neighboring outer (NO) edges in its quarter in order to find a new intersection (see Fig. 7). Push the id of the found edge into the crack array. 3) Jump to the neighboring quarter in which the currently found edge lies and continue to search for a new intersection point by checking neighboring outer, opposing outer (OO) and neighboring inner edges. Push the id of the found edge into the crack array. 4) Follow either step 2 or 3 for any newly found outer or inner intersected edge, respectively.

Instead of simply traversing the edges of each quadrant in any ordering, we follow the edges in an order explained above since this method simplifies having a complete crack array which consists of the intersection points in an order that is easy to follow for triangle generation. For instance, if we have only 3 points in this ar-

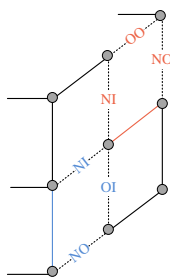


Figure 7: The illustration of an inner (red line) and an outer edge (blue line) together with their neighboring and opposing edges.

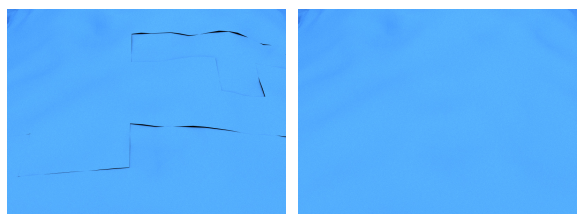


Figure 8: The fluid surface is shown before (left) and after (right) crack handling.

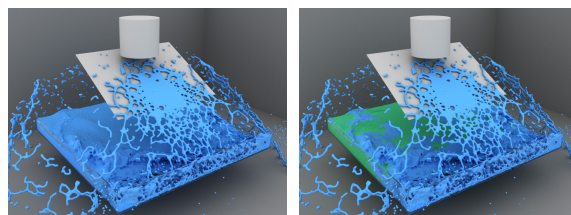


Figure 9: The splash scene with up to 500k particles. The seamlessly stitched mesh is shown on the left; while blue and green regions on the right image shows the high and low resolution parts, respectively.

ray, we simply create a new triangle using these points as triangle corners. If we have four points, then we order the triangles' corners as $c1, c2, c4$ and $c2, c3, c4$ where $c1...c4$ are the entries of the crack array.

In order to prevent redundant computations on intersected edges, we follow the technique discussed by Akinci et al. [AIAT12], in which grid points store each computation information for the corresponding three edges that leaves this point. This technique aims single level uniform grids and we pursue a slightly different approach for our multi level grid. We store the information in separate hash maps for level 2 and level 3 fine cells whose keys are again the grid point *ids*, and data part of each entry is composed of 3 integers that are the intersection points of corresponding edges. We firstly finish the triangulation for both level 2 and level 3 fine cells, and fill their hash maps accordingly. Later, we check the level 2 side of the transition cell faces to see if there is any marked intersection on the corresponding edge. For each transition sub-cell face of the level 3 cell, we check the fine edges of level 2 side and use the information on the associated sub-cell edge. This method is helpful to store the mesh in an indexed format (e.g. Wavefront OBJ). These indexes are also used for an easy retrieval of the corners of the newly generated triangles that fill cracks. At the end of this step, we obtain a surface mesh which contains seamlessly stitched two different resolution mesh blocks (see Fig. 8).

4 RESULTS

In this section, we demonstrate the utility of our approach with two different test scenes. Average per frame timings, the number of level-2, level-3, patch triangles and the memory consumption information of both test scenes can be found in Table 1. The demonstrated surface reconstructions were run single-threaded on Intel Xeon X5680 CPU with 24GB RAM. For the fluid simulation, we employed the PCISPH method [SP09]. For all examples in the paper, we performed rendering using mental ray [mental].

In our first experiment, we show a scene with a rotated plane along which the pouring water scatters (see Fig. 9). The scene was simulated using up to 500k particles.

scene	#particles	t_{sf}	t_{tri}	t_{total}	$\#tri_{l-2}$	$\#tri_{l-3}$	$\#tri_p$	MEM[GB]
Splash	up to 500k	25.5 sec	4.5 sec	30 sec	140k	1.1m	4000	1
Sink	up to 2.5m	18	2	20	295k	76k	587	1.25

Table 1: Average per frame timings in seconds for each scene. t_{sf} , t_{tri} and t_{total} represent the scalar field computation, triangulation and total timing, respectively. $\#tri_{l-2}$, $\#tri_{l-3}$ and $\#tri_p$ show the number of level 2, level 3 and patch triangles in order.

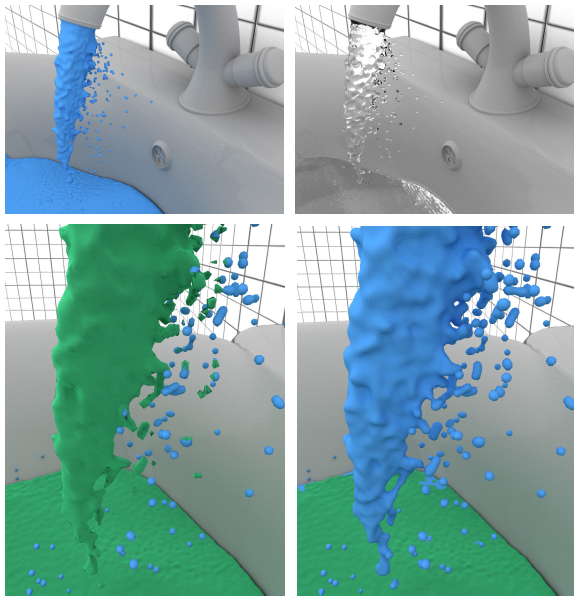


Figure 10: The sink scene with up to 2.5 million particles. Opaque and transparent renderings of the seamlessly stitched mesh are shown in the top row. In the bottom row, results with two different curvature thresholds: 4 (left) and 3 (right) are shown. Green and blue regions represent the low and high resolution parts, respectively.

This test scene shows the increase in high resolution parts and so the computation time (see Table 1) due to the particle scattering and the dominance of high turbulence regions, see Fig. 9-right.

Our next scenario is the Sink scene which consists of up to 2.5 million fluid particles (see Fig. 10 and 11). Using this scene, we firstly experimented with the curvature threshold. The curvature threshold should be chosen by taking the nature of the scene into consideration, e.g., the scene can be turbulent and dominated by high curvature regions or vice versa. Thus, the preferred threshold should guarantee sufficient resolution everywhere. In our experiments, we used 1.5 for the splash scene and 3 for the sink scene. Fig. 10 shows the difference between using two different thresholds on the sink scene. While the value of 3 guarantees a sufficient resolution on high curvature, turbulent regions, e.g. pouring water from the tap; the value of 4 cannot provide sufficient resolution on those parts. Secondly, we compare our method with low and high resolution single level uniform grid approaches which use the cell size of $2r$ and r , respectively. As shown in Fig. 11-left, the low reso-

method	t_{total}	$\#tri_{total}$	MEM[GB]
Uniform low res.	7	310k	1.2
Our method	20	375k	1.25
Uniform high res.	34.5	1.2m	4

Table 2: The comparison of our method with single level uniform grid approaches on the sink scene. t_{total} , $\#tri_{total}$ and MEM[GB] represent average per frame computation time, number of triangles and memory consumption, respectively.

lution single level approach is not able to preserve the details of the pouring water and isolated pieces; and it creates under-tessellated structures. However, it is sufficient to construct the surface properly in the sink where the fluid surface is relatively flat. On the contrary, high resolution single level uniform grid preserves the details properly as can be seen in the right hand side but this resolution is not necessary for the fluid in the sink. Fig. 10 shows that our method uses low resolution inside the sink and high resolution for the pouring water. Finally, we produce a surface which is of similar quality in comparison to the high resolution single level uniform grid (see Fig. 11-middle). Table 2 and Fig. 12 illustrates the test results which show that even we introduce many steps for setting up the grid structure and crack handling, our method runs up to 60% faster than the high resolution single level uniform grid approach with up to 4 times better memory consumption. When compared to low resolution single level uniform grid approach, our approach preserves all of the surface details that yields a proper surface visualization, with an acceptable performance and memory overhead.

When we compare the splash scene and the sink scene, we see that the computation time and the memory consumption is nearly the same for both scenes, even though the sink scene was simulated with larger number of particles. The reason of this similarity is that the particles do not spread around the scene too much in the sink scene in contrast to the splash scene. Therefore, both the coarse grid resolution and the number of surface cells in the narrow band region of the splash scene is slightly larger than the sink scene. Besides, the splash scene is dominated by high resolution parts while the sink scene is dominated by low resolution parts. These points clarify that the computation time depends not only on the resolution of the scene but also on the nature of the scene.

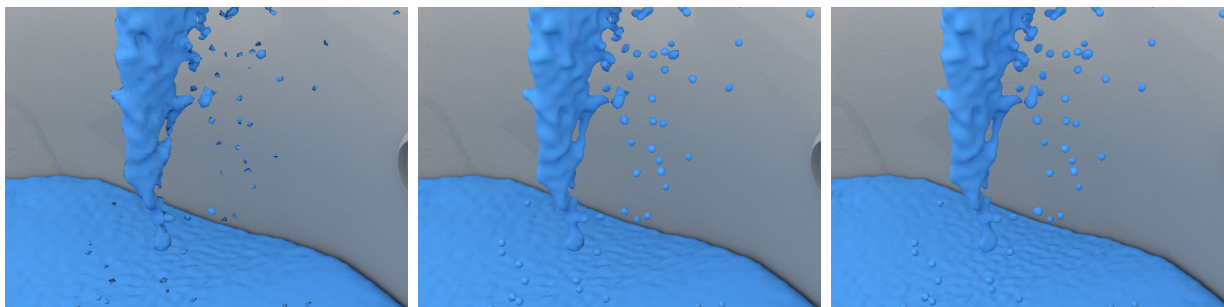
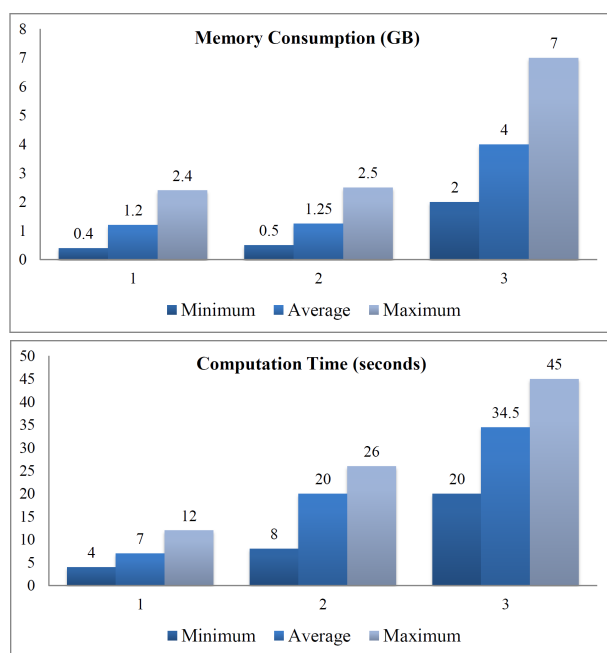


Figure 11: The comparison of the sink scene with single level uniform grid approaches. The low resolution single level uniform grid (left) neither preserve details nor reconstruct the surface properly as our approach (middle) or the high resolution single level uniform grids (right) do. On the other hand, we achieve a similar quality that high resolution single level grids produce with up to four times better memory consumption and up to %60 better performance.



1: Uniform low resolution
2: Adaptive grid
3: Uniform high resolution

Figure 12: Two graphs represent the average per frame memory consumption and the computation time of the Sink scene using our method or high/low resolution uniform grid approaches.

5 CONCLUSION AND FUTURE WORK

In this paper, we presented an adaptive surface reconstruction technique for SPH by using a 3-level uniform grid. The presented grid adapts its cells depending on the surface curvature, which allows for generating less triangles in flat regions but preserving all surface details in curvature regions. Consequently, the number of triangles, memory consumption and the computation time decrease as we produce similar quality results in comparison to high resolution single level uniform grids.

In the future, we would like to consider using hashing for also our top level coarse grid so as to prevent unnecessary data storage for unused cells.

Parallel algorithms have been gaining attention in recent years for high performance computing. However, the incorporation of parallelization in our method is a challenging task due to potential thread safety issues that arise in hashing. Another direction for future work can be addressing these issues and parallelizing our algorithm.

6 ACKNOWLEDGEMENTS

We thank reviewers for their helpful comments. This project is supported by the German Research Foundation (DFG) under contract numbers SFB/TR-8 and TE 632/1-2. The sink model is courtesy of www.turbosquid.com. We also thank NVIDIA ARC GmbH for supporting this work.

7 REFERENCES

- [AIAT12] Akinci, G., Ihmsen, M., Akinci, N. and Teschner, M., Parallel surface reconstruction for particle-based fluids. *Computer Graphics Forum* (2012), 31: pp. 1797-1809.
- [AAIT12] Akinci G., Akinci N., Ihmsen M. and Teschner, M., An efficient surface reconstruction pipeline for particle-based fluids. *Proc. VRI-PHYS, Darmstadt, Germany*, pp. 61-68, Dec. 6-7, 2012.
- [ALD06] Adams B., Lenaerts T., Dutre P.: Particle Splatting: Interactive Rendering of Particle-Based Simulation Data. *Tech. Rep. CW 453*, Katholieke Universiteit Leuven, 2006.
- [BB09] Brochu T. and Bridson R.: Robust Topological Operations for Dynamic Explicit Surfaces. *SIAM Journal on Scientific Computing* 31, 4 (2009), 2472-2493.

- [BGB11] Bhattacharya H., Gao Y. and Bargeil A. W., A Level-set method for skinning animated particle data. In proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA) (2011).
- [Bri03] Bridson R. E., Computational aspects of dynamic surfaces. PhD Thesis, Stanford University. 2003.
- [CPJ10] Costa V., Pereira J. and Jorge J., Multi-Level Hashed Grid Construction Methods. WSCG (2010), Czech Republic.
- [FAW10] Fraedrich R., Auer S., Westermann R.: Efficient High-Quality Volume Rendering of SPH Data. IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2010) (2010), 1533-1540.
- [GSSP10] Goswami P., Schlegel P., Solenthaler B., Pajarola R.: Interactive SPH Simulation and Rendering on the GPU. In Proceedings of the Eurographics / SIGGRAPH Symposium on Computer Animation (SCA) (Aire-la-Ville, Switzerland, Switzerland, 2010), pp. 55-64.
- [HWB04] Houston B., Wiebe M. and Batty C. 2004. RLE sparse level sets. In Proceedings of the SIGGRAPH Conference on Sketches and Applications. ACM.
- [IAAT12] Ihmsen M., Akinci N., Akinci G. and Teschner M., Unified spray, foam and bubbles for particle-based fluids, *The Visual Computer*, vol. 28, no. 6-8, pp. 669-677, 2012, doi:10.1007/s00371-012-0697-9
- [JU06] Ju T. and Udeshi T., Intersection-free contouring of an octree grid. In Proceedings of Pacific Graphics (2006).
- [LD08] Lagae A. and Dutré P., Compact, fast and robust grids for ray tracing. In ACM SIGGRAPH 2008 talks (SIGGRAPH '08). ACM, New York, NY, USA, , Article 20 , 1 pages.
- [LC87] Lorensen W. and Cline H., Marching cubes: A high resolution 3D surface construction algorithm. In SIGGRAPH 1987: Proceedings of the 14th annual conference on Computer graphics and interactive techniques (New York, NY, USA, 1987), ACM Press, pp. 163-169.
- [LY04] Lee H. and Yang H. S., Real-time Marching-cube-based LOD Surface Modeling of 3D Objects, ICAT 2004, 2004-12-00.
- [Len10] Lengyel E., Transition cells for dynamic multiresolution marching cubes. *Journal of Graphics, GPU, and Game Tools* (2010), 15:2, pp. 99-122.
- [Man10] Manson, J. and Schaefer, S., Isosurfaces over simplicial partitions of multiresolution grids. *Computer Graphics Forum* (2010), 29(2): pp. 377-385.
- [mental] Nvidia ARC, 2011. mental ray 3.9 [software]. URL: <http://www.mentalimages.com/products/mental-ray/aboutmental-ray.html>.
- [MSD07] Muller M., Schirm S., Duthaler S.: Screen Space Meshes. In Proceedings of ACM SIGGRAPH / EUROGRAPHICS Symposium on Computer Animation (SCA) (Aire-la-Ville, Switzerland, Switzerland, 2007), pp. 9-15.
- [MCG03] Muller M., Charypar D. and Gross M., Particle-based fluid simulation for interactive applications. In SCA 2003: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 154-159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.
- [Mul09] Muller M.: Fast and robust tracking of fluid surfaces. In Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (New York, NY, USA, 2009), SCA 2009, ACM, pp. 237-245.
- [NM04] Nielsen M. B. and Museth K. Dynamic Tubular Grid: An efficient data structure and algorithms for high resolution level sets. *Linkoping Electronic Articles in Computer and Information Science* 9, 001
- [OR97] Ohlberger M. and Rumpf M., Hierarchical and adaptive visualization on nested grids. *Computing*, 59 (4): 269-285, 1997.
- [RS09] Rosenthal, P. and Linsen, L. (2009), Enclosing Surfaces for Point Clusters Using 3D Discrete Voronoi Diagrams. *Computer Graphics Forum*, 28: 999-1006. doi: 10.1111/j.1467-8659.2009.01448.x
- [SFYC96] Shekhar R., Fayyad E., Yagel R., and Cornhill J. F., Octree-based decimation of marching cubes surfaces. In Proceedings of the 7th conference on Visualization '96 (VIS '96), Roni Yagel and Gregory M. Nielson (Eds.). IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 335-ff..
- [Sigg06] Sigg C., Representation and rendering of implicit surfaces. PhD Thesis, ETH Zurich. 2006.
- [SSP07] Solenthaler B., Schläfli J. and Pajarola R., A unified particle model for fluid-solid interactions. *Computer Animation and Virtual Worlds* 18, 1 (2007), pp. 69-82.
- [SP09] Solenthaler B. and Pajarola R.: Predictive-corrective incompressible SPH. In SIGGRAPH 2009: ACM SIGGRAPH 2009 Papers (New York, NY, USA, 2009), ACM, pp. 1-6.
- [VKSM04] Varadhan G., Krishnan S., Sriram T., and

- Manocha D., Topology preserving surface extraction using adaptive subdivision. In Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing (SGP '04). ACM, New York, NY, USA, pp. 235-244.
- [VT01] Velasco F. and Torres J. C., Cell Octrees: A New Data Structure for Volume Modeling and Visualization. In Proceedings of the Vision Modeling and Visualization Conference 2001 (VMV '01), Thomas Ertl, Bernd Girod, Heinrich Niemann, and Hans-Peter Seidel (Eds.). Aka GmbH, pp. 151-158.
- [vdLGS09] Van Der Laan W. J., Green S., Sainz M.: Screen Space Fluid Rendering with Curvature Flow. In Proceedings of the 2009 symposium on Interactive 3D graphics and games (New York, NY, USA, 2009), ACM, pp. 91-98.
- [WKE99] Westermann R., Kobbelt L. and Ertl T., Real-time exploration of regular volume data by adaptive reconstruction of iso-surfaces. *The Visual Computer* 15 (1999), pp. 100-111.
- [WH94] Witkin A. and Heckbert P.: Using Particles to Sample and Control Implicit Surfaces. In SIGGRAPH 1994: Computer Graphics and Interactive Techniques 28 (New York, NY, USA, 1994), ACM, pp. 269-277.
- [WvG92] Wilhelms J. and Van Gelder A., Octrees for faster isosurface generation. *ACM Trans. Graph.* 11, 3 (July 1992), pp. 201-227.
- [YT10] Yu J. and Turk G., Reconstructing surfaces of particle-based fluids using anisotropic kernels. In Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '10). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp. 217-225.
- [ZB05] Zhu Y. and Bridson R., Animating sand as a fluid. In SIGGRAPH 2005: ACM SIGGRAPH 2005 Papers (New York, NY, USA, 2005), ACM Press, pp. 965-972.
- [ZGHG11] Zhou K., Gong M., Huang X., and Guo B., Data-parallel octrees for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics* 17, 5 (2011), pp. 669-681.
- [ZPvBG01] Zwicker M., Pfister H., Van Baar J., Gross M.: Surface splatting. In SIGGRAPH 2001: Proceedings of the 28th annual conference on Computer graphics and interactive techniques (New York, NY, USA, 2001), ACM Press, pp. 371-378.

Using Image Quality Assessment to Test Rendering Algorithms

Julian Amann

Bastian Weber

Charles A. Wüthrich

CogVis/MMC, Faculty of Media
Bauhaus-University Weimar
Bauhausstrasse 11
99423 Weimar, Germany

{julian.amann|bastian.weber|charles.wuethrich}@uni-weimar.de

ABSTRACT

Testing rendering algorithms is time intensive. New renderings have to be compared to reference renderings whenever a change is introduced into the render system. To speed up the test process, unit testing can be applied. However, detecting differences at the pixel level does not provide a sufficient criterion for the tests. For instance, in the context of games or scientific visualization, we are often faced with random procedurally generated geometry like e.g. particle systems, waving water, plants or molecules. Therefore, a more sophisticated approach than a pixelwise comparison is needed. We propose a Smart Image Quality Assessment algorithm (SIQA) based on a self-organizing map which can handle random scene elements. We compare our method with traditional image quality assessment methods like Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Maps (SSIM). The proposed method helps to prevent the detection of images being categorized wrongly as correct or having errors, and ultimately helps saving time and increases productivity in the context of a test-driven development process for rendering algorithms.

Keywords

image quality, unit testing, computer graphics, self-organizing map

1 INTRODUCTION

During the last three decades, rendering systems have grown immensely in complexity: real time rendering subsystems of 3D gaming engines such as the Unreal Engine can contain millions of lines of code, while offline rendering systems such as RenderMan or Autodesk 3ds Max have been in development for almost thirty years [AGB99]. The increase in complexity of rendering systems makes it a challenging task to ensure at the same time a high software quality.

To handle the increasing software complexity and to assure the quality of such a rendering system, the system must be tested. Testing can be done in different ways. For instance, a quality assurance team can check different outputs of the rendering system and compare them to old output. For example, a scene can be loaded and rendered with the same camera settings and same rendering settings twice: once with an early revision of the

software, and a second time with the new version. If the images generated differ, a software bug or a conceptual error might have sneaked into the rendering system. This manual work is very labor intensive and a tedious task for a tester, especially when many different test scenes have to be checked with all the different variants of render settings.

To improve this situation, unit testing can be used instead of manual testing. Unit testing allows to test the system automatically whenever new code has been introduced into the codebase. Also, a unit test can be executed for instance by a nightly build server to supply the developers daily with a current quality assessment report.

This paper is concerned with the automatization of this testing process, outlines how such an automatic process can be implemented, develops a method for measuring the quality of rendered images and traces the path for managing random output tests for a rendering system.

The novel contribution of this article is a smart image quality assessment algorithm (short SIQA) which can evaluate the quality of an image based on similar reference images. This makes the algorithm unique with respect to existing reference based image quality assessment methods like Mean Square Error measures, Peak

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Signal-to-Noise Ratio (PSNR) or Structural Similarity Index Maps (SSIM) [WBS03].

Our proposed algorithm (SIQA) can be used for black box testing and does not need any knowledge of the internal structure of the render system. This gives the algorithm the typical disadvantages of black box tests, such as being broad based, but also their advantages of being very general and flexible to software changes.

2 MOTIVATION

Software bugs can easily be introduced in a render system when changing its code. Often, bugs are not detected immediately. Sometimes, a bug in a subcomponent of a rendering system can affect other parts of the rendering system. A typical example of such cascading error behaviour can be seen for instance in the discontinuities in the shadows of Figure 1.



Figure 1: A discontinuity in shadow due a software bug.

The growing complexity of rendering systems called for changes in the software engineering process: to ensure a high quality of the images, bugs and errors must be detected early and in a reliable way.

One approach to detect render system bugs early is Test-Driven Development (TDD) [Bec02]. TDD has become quite popular in software development. In TDD tests (unit tests) for a software component are developed before new features are implemented. After the implementation of a new software component, all existing tests are executed to see if one or more of them fail. Quite often, testing fails and early bug tracking will help to produce a better implementation, which eventually will pass all tests defined at the beginning. In the next step of TDD, refactoring is used to eliminate for example redundancies in the source code. This can again introduce new bugs which can be detected early by unit tests.

A test process for a rendering system consists of three steps: at first, the code is modified. Subsequently tests are run. Finally, the test results are analyzed (Figure 2). If a tests fails, the process is restarted. The code is modified again and eventually it will pass all tests.

Using TDD implies a paradigm change in software production: before the introduction of TDD, small ad hoc test applications were made to test the software, but

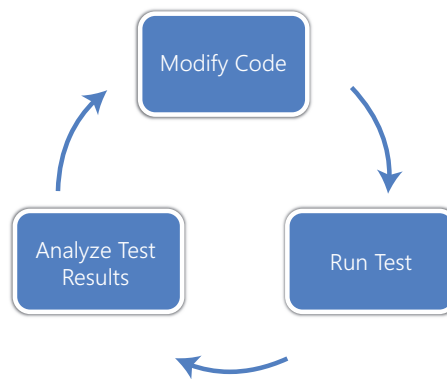


Figure 2: The test procedure

they were discarded after the testing. In TDD, unit tests are as important as production code [Rob08], and code quality of the test code becomes as important as the software itself and gets the same attention and care as the actual software. Testing makes sense even when not using a test driven development process.

2.1 An automatic test process

The TDD steps illustrated in Figure 2 can be easily automated with the help of a test server (e.g. a nightly build server). Developers check in code to a repository. The code contains unit tests that can be executed automatically by a unit test system. The build server automatically checks out the repository on a dialy basis and runs all unit test. Finally, it reports the test results to the developers through a website. This test approach can be implemented for instance with CDash in combination with CTest, CMake, Visual Studio and Mercurial as a version control system [Kit13]. An overview of this test process configuration can be seen in Figure 3.

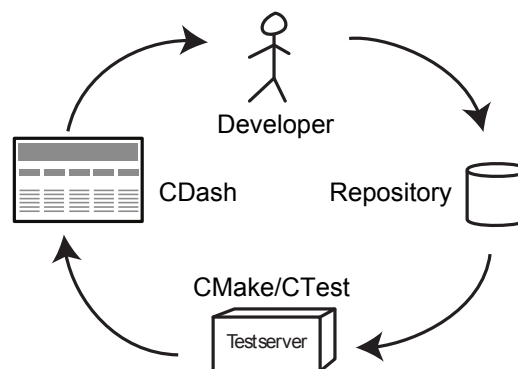


Figure 3: An automated test process with CMake, CTest and CDash

The following pseudo code demonstrates how a unit test is usually implemented:

```
TEST(Core, myAddition)
{
    int result = Bar::myAddition(3, 4);
```



```

EXPECT_TRUE(result == 7);
}

```

The method `myAddition` adds two numbers and returns the computed result. The assertion `EXPECT_TRUE(result == 7)` validates the computed result. It checks if the addition of the numbers 3 and 4 results in the sum of 7. If so, the unit test passes without failures, if not, the unit test framework reports an error.

Because images are difficult to compare, in the context of a rendering system, it is not easy to focus on how to test or what to test on. A trivial solution to this problem could be to generate a reference image, render the scene with the updated software, and compare the reference and the generated image.

For each unit test, a correct reference image (valid screenshot) has to be created. For example, we can create a unit test which checks if tessellation is working. Another test can check if pixel shaders are working or if volume rendering works. Figure 4 shows some examples of such reference images.

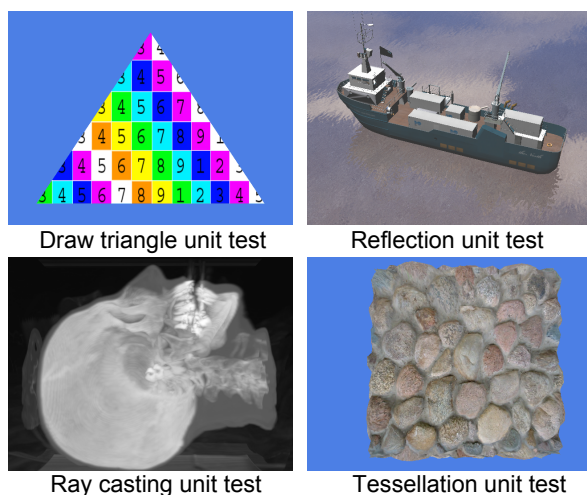
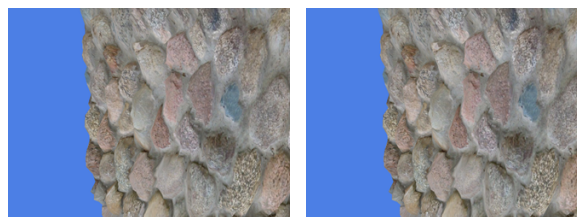


Figure 4: Each picture is a reference image for a specific unit test.

Comparison between the rendered image, the correctness of which is not known, and the reference (a valid screenshot from the past) is done pixel by pixel. If one pixel differs, the test fails. This test would work in case different renderings produce exactly the same output. Unfortunately, this is almost never the case: even the same setting, rendered through two different graphic cards, do not produce the same output. Figure 5 shows renderings of the same scene on a GeForce GTX 480 and on a GeForce GTX 560 Ti: the output is different, but it looks the same at a first glance.

Figure 6 highlights in red different pixels in the rendered output through a GeForce GTX 560 Ti and a GeForce GTX 480. Remarkable here is that both times



(a) GeForce GTX 560 Ti (b) GeForce GTX 480

Figure 5: The same unit test rendered on two different GPUs.

the same program with the same Direct3D 11 render commands and the same driver (GEFORCE R300 DRIVER, 301.42 WHQL) has been executed. This can be due to non-deterministic behavior or implementation defined behavior. The DirectX 11 specification allows some freedom in the concrete driver implementation which can result in such differences. A reason for the different render output can be for instance differences in the texture sampling pipeline across different GPUs.

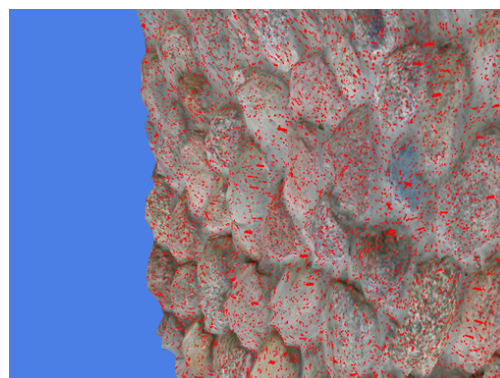


Figure 6: Red pixels mark differences between the render output of a GeForce GTX 560 Ti and a GeForce GTX 480.

A more sophisticated solution to prevent false positives in the classification has to be found. A pixelwise comparison does not work here. Furthermore, a distinction between unintentional randomness in the render output as seen above in the tessellation example and intentional randomness in a scene is necessary. Figure 7 shows two examples of intentional randomness in a scene. The first one is an ivy generator which generates a new ivy structure each time executed. The other example shows waving water. The differences are highlighted in red (Fig. 7(e) and 7(f)). Although it is possible to record the random parameters to regenerate the same scene twice, it can be sometimes quite cumbersome. For example, some data might be only GPU accessible and can be difficult to access from the CPU side. In this case, our technique can help. It is just easier, because it works without knowledge of the internal code structure as a black box test.

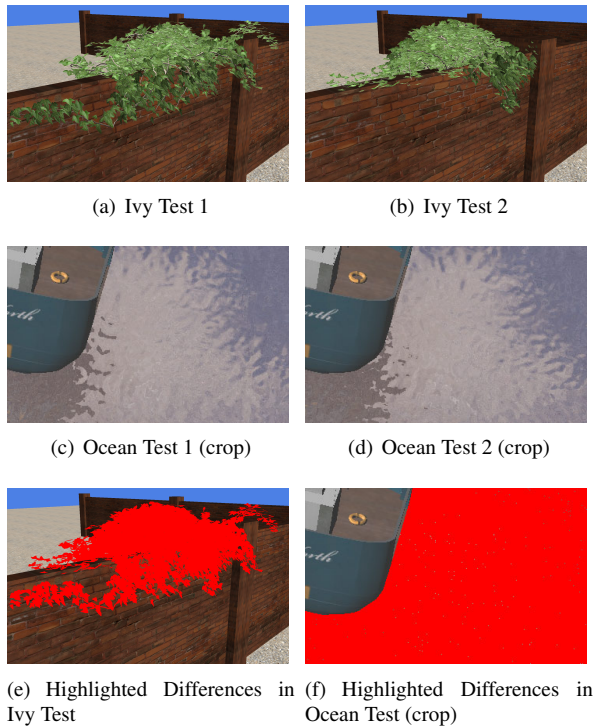


Figure 7: Two examples of intentional randomness

3 RELATED WORK

Quality assessment methods for images can be subdivided into subjective and objective approaches. In subjective assessment, human observers are asked to evaluate the quality of an image, and the results are evaluated statistically. Such methods are slow and expensive, which makes them not well-suited for their use in software development.

Objective methods instead try to determine the quality of an image automatically through some measure. They can be further subdivided into *full-reference*, *no-reference* and *reduced-reference* approaches [WB06], depending on the availability of a reference image, which is assumed to be error free and of highest quality. In reduced reference methods, features are extracted from the reference image and the method only compares this limited set of features with the image the quality of which has to be evaluated, which we will call *subject* image.

One of the most used full-reference methods is the Mean Square Error (MSE), which is the square of the L^2 distance of the images, and computes the square of the difference of corresponding pixels in the reference image A and the subject image B . Let $A = \{a_1, a_2, \dots, a_N\}$ and $B = \{b_1, b_2, \dots, b_N\}$ be the images, then MSE is calculated by:

$$MSE(A, B) = \frac{1}{N} \sum_{i=1}^N (a_i - b_i)^2. \quad (1)$$

MSE provides an overall measure of how the intensities of two images differ from each other [WB06].

Another commonly used measure is Peak Signal-to-Noise ratio (PSNR), which is based on the L^P norm and is defined as follows:

$$PSNR(A, B) = 10 \log_{10} \frac{I^2}{MSE(A, B)}, \quad (2)$$

where I is the maximum allowed pixel intensity (i.e. 255 for an 8-bit grayscale image).

PSNR is also often used to evaluate the quality of an image. A PSNR value is typically in the range of 30 dB to 40 dB. A higher PSNR value is better than a lower PSNR value, since a higher value indicates that the signal is closer to the original signal than a signal with a lower PSNR value [WB06].

MSE and PSNR work only in a full-reference setting and consider an image only on a pixel-by-pixel basis.

A third type of quality measure commonly used in full reference approaches is structural similarity. In contrast to MSE and PSNR, structural similarity does not only consider individual pixels, but also the neighborhood of a pixel. For instance, a structural similarity index can be computed by cropping two small image regions from the reference and subject images and combining the neighborhood by a weighting function. A simple structural similarity index can be computed by considering the luminance and the contrast of an image.

The Structural Similarity Index Map (SSIM) of two image regions \mathbf{x} and \mathbf{y} can be computed by

$$SSIM(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \quad (3)$$

where C_1 and C_2 are constants which prevent a division by zero, μ_x and μ_y are sample means of \mathbf{x} and \mathbf{y} respectively and σ_x^2 , σ_y^2 are the sample variances of \mathbf{x} and \mathbf{y} [LB09]. Usually, the mean SSIM value is used to evaluate the quality of an image.

In a reduced-reference setting, for example keypoint matching can be used. The goal is to find a certain number of interesting points in the reference image and match these in a similar subject image. For instance, the Scale-Invariant Feature Transform (SIFT) can be used here [Low99]. But it is not yet clear how this technique can be used to assess the quality of an image.

[Rob12] introduced a no-reference algorithm for detecting global-illumination based rendering artifacts via a machine learning approach supposedly competitive with state of the art full-reference algorithms. This algorithm uses "right" and "wrong" image samples. After manual masking of the problematic areas, a classifier is trained with features extracted from these areas.

Starting with a pool of features, the most discriminative ones concerning this artifact in consideration are selected. After the detection, a correction method is used, removing the artifacts from the image.

4 SELF ORGANIZING MAPS TO EVALUATE QUALITY

4.1 Overview

Our method is based on a self-organizing map (SOM) which allows us to classify images as correct or wrong. The idea of a SOM, also known as Kohonen map has been introduced in [KT82]. From a high level point of view, a SOM maps feature vectors of an arbitrary n dimensional space to a 2D space. The SOM has the characteristic that feature vectors with similar properties form clusters on the 2D projection of the corresponding SOM.

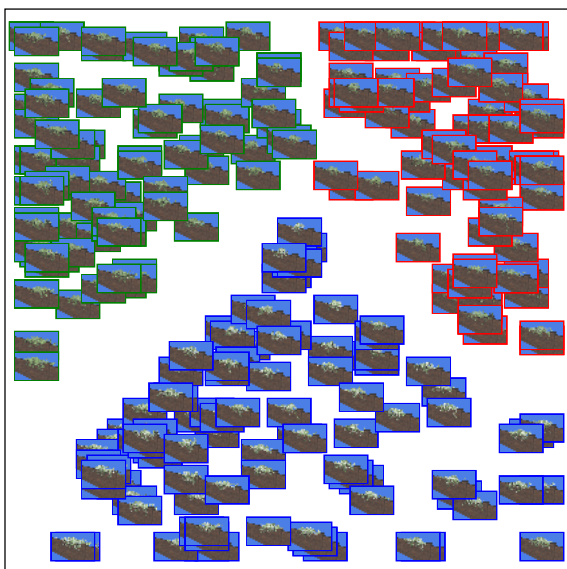


Figure 8: Projection map of classified items on a SOM

Figure 8 shows this effect: as can be seen in the picture, different types of items concentrate to different map regions. Images with a green border are correct. Images with a blue border have incorrect alpha blending issues. A red border marks images with a wrong depth stencil test.

The complicated part of the design of the method for evaluating mistakes in rendering is to find good features that help to distinguish between correct and incorrect images.

Self-organizing maps support a non-supervised learning process. However, to get interesting results one has to pre-classify at least one item. Otherwise, the SOM can only tell us that an image belongs to a certain cluster, for instance to the cluster of the images marked with a red border in image 8. However, it cannot tell

us if the corresponding cluster is a cluster which contains only correct or only incorrect images. Ideally, we pre-classify a set of correct and incorrect images beforehand. A person from the quality assessment team has then to decide if the image is correct or not. The SIQA system stores this information and uses it to make an independent decision for future test images. Furthermore, the SIQA can also compute an estimate of how likely it is correct. This is done by determining the distance of the test image to the already correct classified images in the SOM.

4.1.1 Self-organizing map

Our implementation of a self-organizing map is represented by a square lattice. Each node in this lattice is associated with a vector whose dimension equals the dimension of the feature vector. These are called codebook vectors. At first, the SOM has to be initialized. We implemented random initialization and linear gradient initialization. The latter means that the codebook vectors start with 0 in the upper left corner and increase linearly until the right bottom corner where they reach 1. Afterwards, the training of the SOM may begin. The amount of training feature vectors has to be known to compute the decay of the learning rate and the neighbourhood radius. Additionally, they have to be passed as a set in order to be able to determine the minimum and maximum value of each feature which in turn are used to normalize the values so they are inbetween 0 and 1. This is done automatically during the training, so there is no need for the input to be normalized. The order in which the training vectors are used can in our case either be random or round-robin.

Another option would be to leave the ordering of the items as provided which might result in unpredictable behaviour.

Training then works as follows: the algorithm iterates through all the codebook vectors and calculates the Euclidean distance in feature space between the codebook vector and the training item vector. The item is then projected onto the node with the smallest distance, which is denoted *best matching unit*. Codebook vectors that lie inside the *neighbourhood radius* are modified in a way that they get closer to the feature vector of the projected item. Such radius decreases over time. The starting radius r_{start} is calculated as

$$r_{start} = \frac{w}{2}, \quad (4)$$

where w is the width of the square lattice. The current radius r dependent on the time t is computed as

$$r = \text{decayFct}(r_{start}, t, \tau), \quad (5)$$

while the time constant τ is obtained as

$$\tau = \frac{N}{\ln(r_{start})}, \quad (6)$$

where N is the amount of training items. The decay function is defined as follows:

$$decayFct(a, b, c) = a \cdot e^{-\frac{b}{c}} \quad (7)$$

An additional variable called the learning rate (l) is used in the learning process. This variable is initialized with a user provided value l_{start} which is between 0 and 1. The learning rate controls how much impact the learning process has on the codebook vectors. It decreases over time (t) using the same decay function

$$l = decayFct(l_{start}, t, \tau). \quad (8)$$

The impact of the learning procedure on the nodes inside the neighbourhood radius also depends on the distance to the best matching unit, and decay is defined through a Gaussian bell:

$$gaussFct(\delta, \sigma) = e^{-\frac{\delta^2}{2 \cdot \sigma^2}}. \quad (9)$$

The distance can be calculated as the Euclidean distance using the x,y-coordinates of the nodes. Now with all of the aforementioned parameters, the actual learning process may take place:

$$\psi = \psi + gaussFct(\delta_{bmu}, r) \cdot l \cdot (\omega - \psi). \quad (10)$$

Here ψ is the codebook vector that is getting trained and ω the feature vector used for training. The variable δ_{bmu} denotes the distance between the best matching unit and the node containing ψ . Concerning the time value t , it has to be noted that t is initialized with 0 before starting the training procedure and is increased by 1 after each training item. The order of the training items has a great influence on the quality of the learning. We experimented with random (shuffled) and round-robin order and could achieve the best result with linear gradient initialization in combination with round-robin order.

When the training process is finished, the SOM may be saved as a file, allowing the user to load it again for classification purposes whenever he needs to, saving the training (learning) step to the user.

The classification process is very simple: similarly to the training phase, the best matching unit for an item is chosen. Then, the average distance to all the training items is calculated per item group. The item group with the lowest distance wins and the classification item is classified as that type.

It is also possible to use another type of cluster analysis. [Ran71] describes some objective criteria for the evaluation of clustering methods that can also be considered here. Similar work has also been done in [BGV92].

4.2 Feature Vector

The self-organizing map approach requires feature vectors for training. A feature vector can be seen as an n -dimensional vector, where each component represents one type of feature. Per image one feature vector is created and each feature is described by a floating point value. These values have to be greater than zero. During the training process, they will automatically be normalized between 0 and 1. The purpose of the feature vector is to describe an image as distinct as possible. In short, a number of features is extracted from each image and pooled into a vector.

The task of the self-organizing map is to distinguish different types of images. Therefore, the features must be selected in a way that they are as similar as possible for images of the same kind, and differ as much as possible when images are different. The approach we followed was to design potentially effective features at first and to select those that were actually the most useful afterwards by evaluating them directly on the image set.

4.2.1 Choice of the features

At first, the features chosen were the ones represented in table 1.

For most calculations, only the brightness information of the image is being used. Only the *average hue* and the *color histogram* make use of the color information. Hue, saturation and brightness values were calculated according to the transformations of the HSV color model [Rei08]. Alternatively, also the L*a*b* model could suit the purpose of providing luminosity values. The first block contains values that are extracted from the power spectrum of a 2-dimensional Fourier transform of the image, as shown in Figure 9.

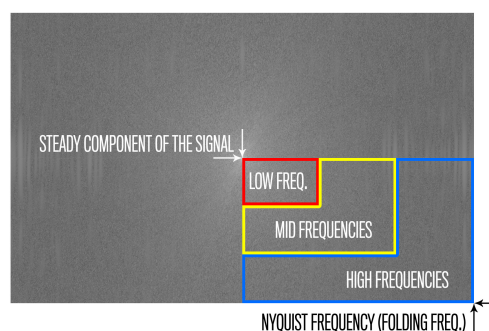


Figure 9: The extraction of the frequency measures

The values of the low, mid and high areas are getting averaged. These values give an impression on how much low, mid and high frequency detail can be found in the luminosity information, which is most important for human perception. The *Pixel-value-median* is a measure for the general brightness of the image. *Edginess* is calculated as the averaged sum of the differences from

Feature group	Feature values
Two-dimensional Fourier transform	Steady component Low frequencies Medium frequencies High frequencies
Medium luminosity	Median of the pixel values
Horizontal and vertical edginess	Summed absolute differences in horizontal direction Summed absolute differences in vertical direction
Average hue	Average of the HSV hue value
Darkest and brightest areas	Darkest pixel x-Coordinate median Darkest pixel y-Coordinate median Brightest pixel x-Coordinate median Brightest pixel y-Coordinate median Darkest pixel value Brightest pixel value Darkest pixel amount Brightest pixel amount
Average gradient	Average gradient x-direction Average gradient y-direction Average gradient length
Histogram	Red channel histogram Green channel histogram Blue channel histogram

Table 1: The features

one pixel to the next, once in the horizontal and once in the vertical direction. *Average hue* is obtained by averaging the hue information of all pixels, providing a value for the prevalent color in the image. Darkest and brightest pixels denote the pixels with the highest brightness value and the pixels with lowest brightness value in the image. There might be one or more pixels of that kind, the exact number is saved in the corresponding features *darkest pixel amount* and *brightest pixel amount*. The median of the x- and y-coordinates of these pixels is also being used as a feature, indicating the position where the brightest and darkest areas in the image are concentrated. *Darkest and brightest pixel value* denote the exact brightness values of the brightest and darkest pixel. This feature group gives a hint on how these very distinctive areas are distributed over the image. The *gradient* is a 2-dimensional vector that is based on the gradients in x- and in y-direction. These gradients are calculated by making use of the difference quotient

$$f'_{central}(x_i) = \frac{f(x_{i+1}) - f(x_{i-1}))}{2}. \quad (11)$$

The x- and y-gradients are combined into a vector. Then, all gradient vectors are averaged, while the direction is only taken into consideration, if the gradient length lies beyond a certain threshold. The direction, as well as the length of this average vector, is being used as a feature providing information of the edge distribution in the image. The *color histogram* is defined through the red, green and blue component histogram. Each of those is generated out of the red, green and blue channels of the image by calculating the relative occurrence frequency of each color value, meaning the percentage of pixels of this color in the respective channel. The amount of histogram values is reduced by combining a certain amount of neighbouring values through averaging. This way, a histogram width of 32 values is reached. The color histogram can reveal changes in the color distribution.

4.2.2 Evaluating Feature Effectiveness

As mentioned above, an evaluation of the features was performed directly on the image set that should be worked on. The process of this evaluation was implemented in the following way: assuming we have m groups of images denoted as G_i with one feature vector v_j per image

$$G_i = \{v_1, v_2, v_3, \dots, v_p\}, \quad (12)$$

and that these feature vectors all contain n features,

$$v_j = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_n \end{pmatrix}, \quad (13)$$

then the average α and variance σ of the feature k in group i is given by

$$\alpha(k, G_i) = \frac{1}{|G_i|} \sum_{j=1}^p v_j[k] \quad (14)$$

$$\sigma(k, G_i) = \frac{1}{|G_i|} \sum_{j=1}^p (v_j[k] - \alpha(k, G_i))^2. \quad (15)$$

Now the distance between two groups G_h and G_i concerning the feature k can be computed as

$$\delta(k, G_h, G_i) = |\alpha(k, G_h) - \alpha(k, G_i)|. \quad (16)$$

$\delta(k, G_h, G_i)$ indicates how well the feature k is suited for distinguishing items of the groups G_h and G_i .

A more complete approach for selecting the best features could be done through principal component analysis. However, this is beyond the scope of this paper.

5 RESULTS

There are tasks where conventional methods like MSE or SSI are completely sufficient for quality assessment. One is, as turns out, the evaluation of defective images as shown in Figure 5. The error is small enough to be able to discriminate correct looking images that only contain minor errors caused by differences in driver implementation (referred to as false negatives) from actually erroneous images (referred to as correct rejections) by defining a threshold value for MSE or SSIM. Table 2 shows this: for the false negatives, the SSI value is always very close to 1, while for the correct rejections the values are lower than 0.8. Hence, it would be possible to say that all images with an SSI value above 0.99 are supposed to be seen by an observer as being correct. The same conclusion can be deduced from the MSE values. They are all below 0.05 for the false negatives and above 3000 for the correct rejections.

False Negatives		Correct Rejections	
MSE	SSI	MSE	SSI
0.0490115	0.999995	18951.6	6.52E-14
0.000315755	1	3248.31	0.748823
0.031875	0.999983	10301.5	1.04E-07
0.00200065	1	10699.3	1.60E-07
0.000706706	1	10301.5	1.04E-07
0.000199653	1	19775.1	0.754946
<0.05	>>0.99	>3000	<0.8

Table 2: MSE and SSI values for images with minor (false negatives) and major errors (correct rejections)

Figure 10 shows an example of randomly generated plants. While 10(a) shows a correctly rendered version, 10(b) (referred to as Error-1) and 10(c) (referred to as Error-2) show two kinds of rendering errors.

The images look different each time they are rendered, since the plants are based on a stochastic approach. Mean squared error (MSE) and Structural Similarity Index (SSI) measures are not suitable here due to the fact that there is no reference image that can be used for comparison. To prove this, we classified images through MSE and SSI. As footage, we used 300 of the ivy images, 100 for each type. We then compared each image with 30 images of each group and averaged the MSE respectively SSI value per group. The image was then classified as the type that had the lowest average MSE value or in case of SSI the highest value. The MSE results can be seen in table 3, and the SSI results are shown in table 4.

	Classified as		
	Correct	Error-1	Error-2
Correct	52	48	0
Error-1	12	88	0
Error-2	9	91	0

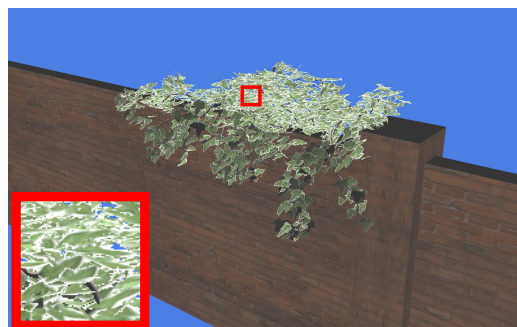
Table 3: Classification via MSE



(a) Reference



(b) Depth ordering issue



(c) Alpha blending issue

Figure 10: Generic plants

	Classified as		
	Correct	Error-1	Error-2
Correct	54	46	0
Error-1	49	51	0
Error-2	79	10	11

Table 4: Classification via SSI

As can be seen, results are not satisfying. In the case of MSE, no image at all was classified as Error-2 type. The rest is distributed randomly, only for Error-1 a small correlation in the case of MSE could be seen. SSI performs even worse, although it classifies 11 images as Error-2.

We then trained a self-organizing map with 100 images of each kind, using linear gradient initialization and round-robin order. Afterwards, we classified the same set with the trained SOM. Figure 8 shows the result of the classification. The red, blue and green frames rep-

represent the different image types. It turns out that they are concentrated in three different areas of the SOM and three distinct clusters can be observed. The classification success in numbers can be seen in table 5.

	Classified as		
	Correct	Error-1	Error-2
Correct	100	0	0
Error-1	0	100	0
Error-2	0	1	99

Table 5: The results of the SOM classification

The amount of false negatives, meaning images that are correct but have been classified as erroneous, in comparison between the different evaluation methods pixel-wise comparison (PWC), Mean Squared Error (MSE), Structural Similarity Index (SSI) and our Smart Image Quality Assessment Algorithm (SIQA), is shown in table 6.

	PWC	MSE	SSI	SIQA
False Negatives	100%	48%	46%	0%

Table 6: Comparison of the evaluation methods

The results of the SOM are a big improvement compared to MSE or SSI. Nearly all images were classified correctly, only *one* Error-2 image was classified as Error-1. Table 7 shows how the classification success increases, depending on the amount of items used for training the map. Here, the set was split 70:30 (classification:training). We then started with one training item per group and increased the amount gradually. Just five training images per set were enough to level down the false negatives and false positives to zero.

Training Item Count	1	2	3	4	5
False Positives	41%	9%	1%	1%	0%
False Negatives	1%	1%	0%	1%	0%

Table 7: False positive and false negative rate in correlation to the amount of training items (per set).

For practical purposes, it would only be necessary to distinguish between correct and incorrect images, which would further lower the complexity of the decision. Naturally, the quality of the results depends on the used images, the selection of the features and the quality of the SOM implementation. Hence, it would probably be unrealistic to expect equal good results in all cases. But the example shows the potential of this approach.

Concerning the performance of the algorithm, it can be stated that the runtime is linear with respect to the amount of items used and quadratically with respect to the width of the lattice (see tables 8 and 9). On a 2.4 GHz dual-core mobile CPU with 8 GB RAM training and classification both took approximately 5.9 ms per item using a lattice width of 100. Feature extraction takes most time: approximately 600 ms per image

(1600x1000 pixels). It has to be noted that the feature extraction was not optimized for performance in any way and that the code was not multi-threaded.

Width (px)	32	64	128	256	512
Duration (s)	0.183	0.291	1.006	4.068	15.864

Table 8: The performance dependent on the width of the lattice

Items	8	16	32	64	128
Duration (s)	0.75	0.142	0.255	0.465	0.96

Table 9: The performance dependent on the amount of items (training)

A possible software production workflow using quality assessment could look as follows: a number of scenes is chosen, possibly using all critical parts of the rendering engine. Two pools of images are created, one with correctly rendered images, and one with images containing artifacts or errors. For each scene, the best features are selected, a SOM is trained and saved as file. Each time the code has to be assessed, an amount of images is rendered for each scene, the corresponding SOM lattice gets loaded and the images are classified. Even if one might not want to fully rely on automatic evaluation, this can give at least a good hint on whether the images are correct or not.

6 CONCLUSION AND FUTURE WORK

In this paper we proposed the use of image quality assessment methods for automatically testing rendering software. The use of automatic quality assessment can ease the rendering software development process, since errors in software development can be discovered early and do not remain undetected until the final version of the software is released. Depending on the characteristics of the scene to be rendered, traditional assessment methods such as MSE, PSNR and SSIM measures perform differently. They are quite helpful for small detail differences, but deliver wrong results if procedural or randomized algorithms are used in the rendering pipeline.

In the latter case, using Self Organizing Maps can compensate and automatically recognize and categorize errors. After a first training phase, SOMs deliver quite accurate results and allow to automatically warn if the image being generated by the software has problems. Of course, the methods proposed for quality assessment have still to be expanded and tuned for their application in real, less controlled situations. In other words, their applicability under real life circumstances has to be thoroughly tested: standard image sets with a variety of images containing artifacts and rendering errors of different types have to be developed. The self organizing map should then be trained with a selection

of erroneous images and, of course, with correct images. Then, the classification must take place with images containing errors that did *not* appear in the images used for training. Only this way a realistic estimation of whether this method is really suitable for practical use is possible. The implementation of the self organizing map still offers great potential for improvements, considering the ordering of the training items, the initialization of the map prior to training and the training process itself.

The selection of the features is a factor with a lot of potential for improvement. As mentioned before, there are alternative approaches for the discrimination of the really useful features with respect to a specific scene, like principal component analysis.

Also, instead of a SOM, other types of neural networks could be used and be tested for their suitability in image quality assessment. For instance, multilayered networks in conjunction with the back-propagation algorithm can be considered [Roj96]. There is also a wide diversity of learning methods like for instance Quick-prop and network types like the Hopfield Model which can also be considered. Furthermore, variants of SSIM like Multi-scale SSIM, Modified gradient-based SSIM [LB09] or Complex Wavelet Domain Structural Similarity Map [WB06] [CW12] could be tested.

As a conclusion, this paper is a simple start of a new theme which is worth exploring, since it bears a lot of potential to ease hardware and software development for Computer Graphics.

7 REFERENCES

- [AGB99] Apodaca, Anthony A. and Gritz, Larry. Advanced RenderMan: Creating CGI for Motion Picture, Morgan Kaufmann Publishers Inc., 1999.
- [Bec02] Kent Beck. Test Driven Development: By Example, Addison-Wesley Longman Publishing Co., Inc., 2002.
- [BGV92] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In Proceedings of the fifth annual workshop on Computational learning theory (COLT '92). ACM, New York, NY, USA, 144-152. 1992.
- [CW12] Jiayin Chen and Charles A. Wuethrich. Analog film, digital images and sampling grids: a Fourier and Wavelet analysis. In D. Thalmann, K. Zhou, CGI 12, Proceedings of Computer Graphics International 2012, Bournemouth, UK, July 2012.
- [Kit13] CDash, <http://www.cdash.org/>, Kitware, 2013.
- [KT82] Kohonen, Teuvo. Self-Organized Formation of Topologically Correct Feature Maps. Biological Cybernetics 43 (1): 59-69. 1982.
- [LB09] Li, Chaofeng and Bovik, Alan C., Three-component weighted structural similarity index, Proc. SPIE 7242, Image Quality and System Performance VI, 72420Q, 2009.
- [Low99] Lowe, David G., Object recognition from local scale-invariant features, Proceedings of the International Conference on Computer Vision. 2. pp. 1150-1157. 1999.
- [Ran71] William M. Rand, Objective Criteria for the Evaluation of Clustering Methods, Journal of the American Statistical Association, Vol. 66, No. 336, 1971.
- [Rei08] Erit Reinhard, Erum Arif Khan, Ahmet Oguz Akyuz, Garret M. Johnson, Color Imaging: Fundamentals and Applications. A K Peters, Ltd., 2008.
- [Rob08] Martin, Robert C., Clean Code: A Handbook of Agile Software Craftsmanship, Prentice Hall PTR, 2008.
- [Rob12] Robert Herzog, Martin Cadik, Tunc O. Aydin, Kwang In Kim, Karol Myszkowski and Hans-P. Seidel, NoRM: No-Reference Image Quality Metric for Realistic Image Synthesis, The Eurographics Association and Blackwell Publishing Ltd., 2012.
- [Roj96] Rojas, R., Neural Networks: a systematic introduction] Neural Networks: a systematic introduction, Springer, 1996.
- [WB06] ZhouWang and Alan C. Bovik, Modern Image Quality Assesment. Morgan & Claypool Publishers, 2006.
- [WBS03] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli, Image quality assesment: From error visibility to structural similarity. IEEE Trans. Image Processing, 13(4):600C612, Apr. 2004.

Handling haptics as a stand-alone medium

Konstantinos Moustakas
University of Patras,
Electrical and Computer Engineering Department
University Campus, Rion
Greece 26504, Rion, Patras
moustakas@ece.upatras.gr

ABSTRACT

This paper provides an information theoretic perspective of haptic rendering that aims to formulate the necessary theoretical foundations to lead to a compact and holistic representation of haptic media. Initially, an information theoretic view on haptic rendering is presented that enables the development of metrics enabling measuring of haptic information. Following the principles of information theory, two novel haptic information rendering pipelines are provided. Moreover, several quantities like entropy, mutual information and filters are defined and instantiated for the case of haptic rendering. Explicit examples of their potential use are analyzed. The paper concludes with examples and a discussion on how information theory can be used to provide a holistic haptic media representation scheme that can be used in a variety of application including coding, indexing and rendering of stand-alone haptic media.

Keywords

Haptic rendering, information theory, haptic coding, indexing.

1 INTRODUCTION

As information and communication technologies become more mature the amount of information generated and exchanged exceeds every previously imaginable limit. While this amount of information is expected to double each year, it is mainly comprised of media information [Chinchor10] including images, videos, sounds, etc. From a holistic media perspective, their historical evolution clearly demonstrates an increment in the dimensionality of the underlying media sources. Starting from two-dimensional still images, moving to the three-dimensional moving picture, further adding one and two dimensions of mono- and stereo-sound respectively and recently also including the sense of depth in stereoscopic visual representations, currently available and widespread media can be considered as six dimensional. Even if more dimensions can be assumed for different features like color, etc. they are considered as features of the 6D space.

The aforementioned increment in the dimensionality of media has been quite remarkable leading to impressive and immersive media. However, new dimensions will

be inevitably added in the media of the future, possibly including digital representations of signals triggering the human senses, that have not been addressed so far in a systematic way, like touch, olfaction and taste. While all three senses could provide a significant added value in the realism and quality of the future media, research in the haptics domain has progressed significantly to allow for an attempt of systematic and formal definition, representation, processing and rendering of the underlying haptic media.

Figure 1, illustrates the complex issue of media source and interrelation, from an information source perspective. On the left part the physical environment is the source of all “direct” media. In particular, light interactions, pressure oscillations and collisions/interaction are considered as the source of “direct visual”, “direct audio” and “direct haptic” information respectively. It is evident that all direct media are correlated, since they are due to the same source even if they are caused by different interactions. Moreover, the “media space” is augmented by auxiliary information that is characterized as “symbolic”. For example “symbolic visual” information includes some forms of visual effects, overlays or even augmented reality renderings; “symbolic audio” is used to describe audio effects or music, while term “symbolic haptic” information is used to describe the synthetic haptic information, including haptic icons [MacLean08]. It should be emphasized that even if the symbolic media are not correlated per se, they could exhibit some correlation depending on the particular case.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

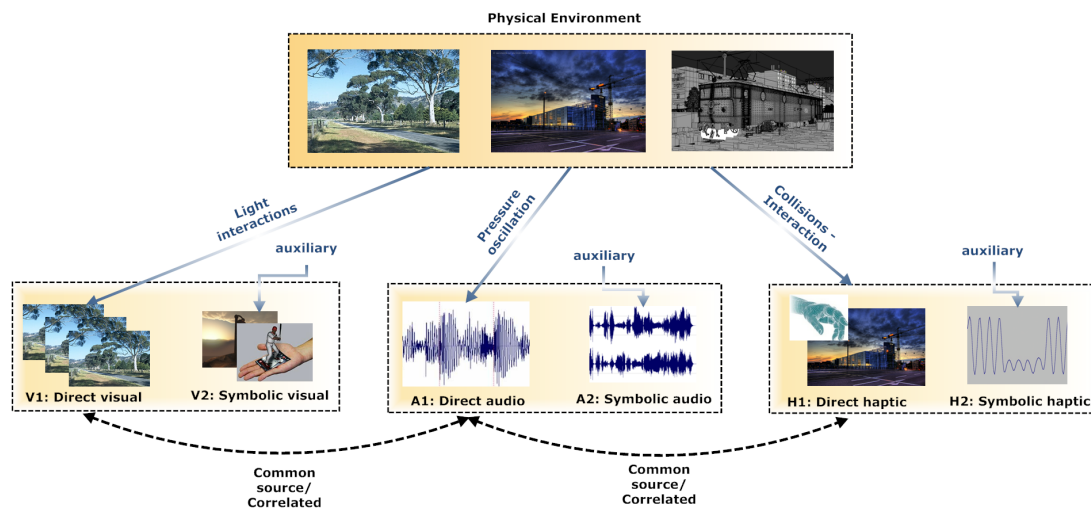


Figure 1: View of haptics as a medium. Left: Physical environment is the source of all “direct” media. Light interactions, pressure oscillations and collisions/interaction are considered as the source of “direct visual”, “direct audio” and “direct haptic” information respectively. Auxiliary symbolic media can be added for all communication channels. “Symbolic visual” information includes some forms of visual effects, “symbolic audio” is used to describe audio effects or music, while term “symbolic haptic” information is used to describe the synthetic haptic information, including haptic icons.

The proposed paper makes a first attempt to define a theoretical framework of haptic media from an information theoretic perspective that could be in turn used as a mathematical background for several processes including representation, processing, rendering and streaming transmission of haptic media.

1.1 Related work

Human perception combines information of various sensors, including visual, aural, haptic, olfactory, etc., in order to perceive the environment. A very descriptive analysis on the importance of the sense of touch is given in [Torre06] through the effects of its loss that include inability to eat and walk. On contrary to the audio and vision channels, the haptic media have not yet been addressed systematically from an information theoretic perspective, leading to non-holistic, fragmented and problem-targeted research.

Haptics research can be divided into three main categories [Lin08]: Machine Haptics, Human Haptics and Computer Haptics [Srinivasan97]. Machine Haptics is related to the design of haptic devices and interfaces, while Human Haptics is devoted to the study of the human perceptual abilities related to the sense of touch. Computer Haptics, or alternatively haptic rendering, studies the artificial generation and rendering of haptic stimuli for the human user. It should be mentioned that the proposed framework takes into account recent research on human haptics, while it provides mathematical tools targeting mainly the area of computer haptics.

The simplest haptic rendering approaches focus on the interaction with the virtual environment using a single point [Moustakas06], [Moustakas07b], [Kaklanis09], [Kaklanis10]. Many approaches have been proposed so far both for polygonal, non-polygonal models, or even for the artificial generation of surface effects like stiffness, texture or friction [Moustakas07], [Moustakas07b], [Laycock07], [Kostopoulos07], [Nikolakis06]. The assumption, however, of a single interaction point limits the realism of haptic interaction since it is contradictory to the rendering of more complex effects like torque. On contrary, multipoint, or object based haptic rendering approaches use a particular virtual object to interact with the environment and therefore, besides the position of the object, its orientation becomes critical for the rendering of torques [Laycock07], [Nikolakis06]. Apart from techniques for polygonal and non-polygonal models [Laycock07], voxel based approaches for haptic rendering [Petersik01] including volumetric haptic rendering schemes [Palmerius08] have lately emerged. Additionally, research has also tackled with partial success the problem of haptic rendering of dynamic systems like deformable models and fluids [Barbic09], [Cirio11].

From information theoretic perspective, it is worth mentioning that surprisingly coding and compression of haptic data has not been researched extensively so far. Most of the approaches deal with aspects of haptic data transmission in the context of telepresence and teleaction systems, focusing mainly on stability

and latency issues [Hirche05], [Ou02], [Souayed03], [Kron04]. Differential and entropy coding has been successfully applied in [Kron04], while other traditional approaches including, DPCM ADPCM, Huffman coding have been applied to haptic signals in [Kron04], [Ortega02] and [Hikichi01] respectively. In [Borst05] predictive coding has been applied for haptic signals in order to optimize the communication in teleaction systems with respect to sampling rate, while in [Hinterseer08], [Kuschel09], [Zadeh08], [Vittorias09] perceptual coding mechanisms are developed for haptic data streams. The major limitation of the current approaches is that, since they are targeting mainly telepresence and teleaction systems, they refer mainly to representation and coding of single haptic timeseries that correspond to the force that should exert a remotely manipulated device.

1.2 Motivation and contributions

In general, with the exception of some approaches related to haptic rendering of distance or force fields [Moustakas07], [Barlit07], one of the biggest limitations of current schemes is that haptic rendering is considered only as a result of the interaction of a human user with an underlying (usually 3D) environment. Even if this approach is inspired by the real world, where the sense of touch is triggered by collisions of the human body with the environment, it has three significant drawbacks: i) haptic media cannot be defined as a signal related to a specific physical environment, ii) Off-line processing of haptic media for optimization, compression, indexing, etc. becomes impossible, iii) The requirement for 1kHz update rate, that is considered as the most significant constraining factor imposed on haptic rendering algorithms [Laycock07], can rarely be satisfied due to the need for on-line processing of the entire haptic rendering pipeline as also mentioned in a high-level theoretical attempt on haptic media using the existing concepts on haptic interaction described in [Cha09].

The proposed framework aims to provide a new view on haptic rendering from an information theoretic perspective that will in turn provide the necessary theoretical foundations to lead to a compact and holistic representation of haptic media. As a result all three aforementioned limitations will be theoretically overcome providing a new potential for the field of haptic rendering. In particular, the proposed approach does not restrict its framework on interaction-based rendering, but provides a holistic approach, that can also accommodate synthetic authored haptic media, where interaction-based rendering is a special case.

The rest of the paper is organized as follows. Section 2 outlines the links between information theory and haptic rendering and presents three haptic render-

ing pipelines. In Section 3 well known information theoretic quantities are introduced and instantiated from a haptic rendering perspective, while Section 4 presents how device and perceptual haptic filters could be designed and applied in the context of the proposed framework. Finally, Section 5 discusses potential applications and open problems for future work, while conclusions are drawn in Section 6

2 INFORMATION THEORY AND HAPTIC RENDERING

The typical models of a communication system has been used as a basis to develop information theory [Shannon48]. Following a similar methodology, this section describes how haptic rendering can be seen as a communication channel and how typical quantities of information theory can be adopted to more efficiently describe haptics as a medium.

Figure 2a illustrates a typical communication system; The source is generating data that are compressed in the “encoder” and transmitted as a signal through the communication channel. At the receiver the signal is decoded and the resulting data is provided to the destination for rendering or processing. It should be emphasized that the diagrams of Figure 2 refer only to source coding. Even if it is possible to derive correspondences between channel coding and haptic rendering as well, this issue remains out of the scope of the present paper.

Figure 2c illustrates a general block diagram that is used in many recently proposed systems for coding and transmission of haptic information. In particular based on traditional collision and reaction models, the force feedback of the interaction of a specific haptic-probe with the environment is initially estimated. Then the force is filtered, usually to avoid force discontinuities, force effects are added, including texture or friction, while the final resulting force is mapped onto the specific display capabilities, e.g. resolution, workspace. Then the resulting time-series is transmitted using traditional coding-decoding approaches and rendered at the receiver. The major characteristic of this approach, called so forth “soft HR system”, is the fact that the force at the transmitter side at each time-step is calculated based on the actions of the end-user at the receiver side. This issue reduces from the one side the amount of haptic information to be transmitted but on the other side, introduces latency issues and places severe limitations when targeting at explicit and stand-alone haptic media that can be processed and rendered as a single entity, without the need to know the user actions beforehand.

Towards this target, Figure 2b depicts a general potential “strong HR system” that implies that haptic information is available as a multivariate time-series that

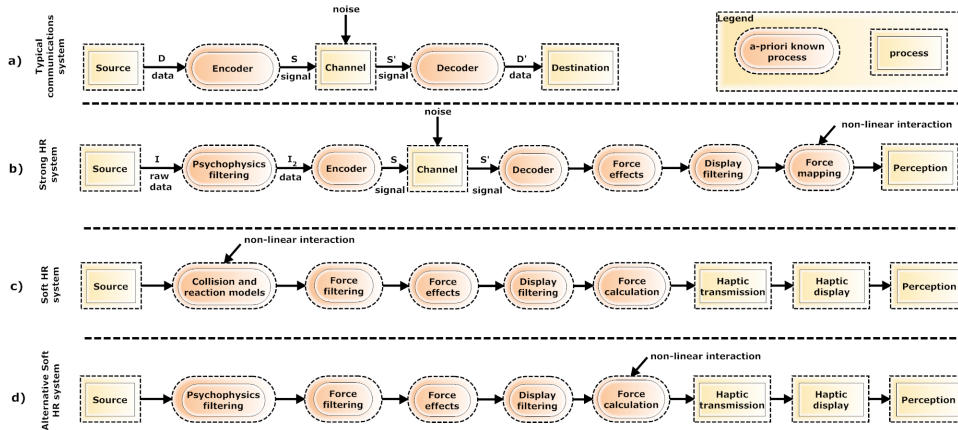


Figure 2: Haptic rendering from an information theoretic perspective. a) Typical communication system, b) Strong haptic rendering (HR) system: Haptic media are encoded in a similar manner to Figure 2a. Since force mapping is performed at the decoder it is device agnostic, c) Soft HR system: Is based on interaction-based haptic rendering; block “Collision and reaction models” is influenced by the end user actions and haptic information is transmitted only for the necessary degrees of freedom that are rendered, d) Alternative Soft HR System: Follows the approach of Figure 2b placing however all haptic rendering processes at the encoder side, thus being device specific.

needs to be compressed and transmitted, after applying psychophysics filtering, through a communication channel. At the decoder the raw data are recovered and enriched with force effects and mapped for rendering on a specific display. This scheme has the advantage that haptic media is actually encoded as a time-varying, device agnostic vector field that can be used by any kind of haptic rendering approach. Special effects and force mapping are added at the receiver side according to potential metadata sent by the transmitter or even based on the preferences of the receiver.

Similarly, Figure 2d introduces an alternative to Figure 2b and Figure 2c communication system that tries to capture some interesting properties of both sides. Actually the main difference between the “Alternative Soft HR system” and the “Strong HR system” is that it places the force effects and display filtering and mapping at the transmitter side. This approach still treats haptic information as a stand-alone entity, while it also reduces the amount of information to be transmitted with respect to the “strong HR system”. On the other side, it is less general since different receivers would require different streams to be transmitted, which is not ideal if haptic media would need to be transmitted in a broadcasting sense.

In the following, the potential use of several information theoretic quantities for the case of haptic media is described, that can be applied for both the “Strong HR system” and the “Alternative Soft HR system”.

3 MEASURING HAPTIC INFORMATION

In the following the major information theoretic quantities, namely entropy and mutual information, that will

be used in the subsequent analysis will be described and adapted for the case of haptic media.

3.1 Entropy

Entropy is maybe the fundamental measure in information theory. It can be seen either as the uncertainty of a given random variable or as the theoretical minimum number of bits that are required to represent the variable.

The entropy of a discrete random variable \mathbf{F} that takes values in an alphabet \mathbf{A} and has a probability mass function $p_{\mathbf{F}}(f)$ is given by

$$H(\mathbf{F}) = - \sum_{f \in \mathbf{A}} p_{\mathbf{F}}(f) \log_2 p_{\mathbf{F}}(f) \quad (1)$$

Let us now consider, without loss of generality, a haptic workspace of $5 \times 5 \times 5$ voxels. For each point in space in between the voxel centres, tri-linear interpolation is used to render the force if necessary. Moreover let us also assume that the force that is attributed to each voxel is quantized and takes values from an alphabet of size 256. The probability mass function of each voxel is independent and identically distributed, so there is $p = 1/2^8$. Now letting V denote the random variable for a single voxel and S denote the full workspace, the total entropy is calculated as follows:

$$H(S) = \sum_{i=1}^{125} H(V_i) = - \sum_{i=1}^{125} \sum_{j=1}^{256} \frac{1}{2^8} \log_2 \frac{1}{2^8} = 1000 \quad (2)$$

The above result is theoretically expected since for each voxel 1 byte is needed to represent the force value and there are 125 voxels in the workspace.

Haptic workspace capacity:

It is reasonable to assume that in most applications the reference space or virtual space is of different size compared to the haptic workspace. Let us now assume a mapping M of the haptic information of the actual space into haptic information of the haptic workspace. The latter maybe restricted by several parameters including haptic display limitation on the workspace size, resolution, force exertion amplitude limitations, arithmetics, etc. Let us now define as Workspace Capacity $C(M)$ the average amount of information that the specific haptic workspace setting can render. Since the above parameters are usually constant $C(M)$ is the entropy of a random variable of the specific mapping M . For a specific input haptic space W , following the principle of the data processing inequality the following equation holds:

$$C(M) = \min(C(M(W)), H(W)) \quad (3)$$

where the *min* function encodes the fact the mapping is passive, i.e. no energy/information can be generated through the mapping procedure, e.g. in the typically not usual case where the haptic information space is less detailed than the haptic workspace itself.

Since the quantity C has the same unit as entropy H , we can define the following measures:

$$\text{Workspace Mapping Ratio (WMR)} = \frac{C(M)}{H(W)} \quad (4)$$

$$\text{Haptic Inf. Loss (HIL)} = \frac{\max(H(W) - C(M), 0)}{H(W)} \quad (5)$$

The above measures are meaningful under the reasonable constraints $H(W) > 0$ and $C(M) > 0$. The WMR reflects the amount of information that is transferred for haptic rendering through the haptic workspace, while the HIL encodes the amount of information that is lost in the above procedure.

3.2 Joint Entropy and Mutual Information

The fact that haptic interaction involves in general several computationally intensive procedures like 3D mesh manipulation leads researchers in the adoption of a level-of-detail (LoD) haptic rendering scheme. This means that each object or space partitioning element can be rendered in different scales. For example a user might need initially to interact with the complete environment and then zoom in a specific area so as to analyze it in more detail and in higher resolution.

In information theoretic terms this case can be described through different mappings M . In particular there is one mapping related to the whole scene M_{all} and several LoD mappings $M_{LOD(i)}$. Obviously M_{all} and $M_{LOD(i)}$ are related.

Let us now assume two random variables X and Y that represent M_{all} and $M_{LOD(i)}$ respectively, assuming that the latter act as mappings on the input space Z shared by both full and LoD representations. Let also $H(X, Y)$ represents their joint entropy that is defined through their joint pmf $p_{X,Y}(x, y)$, $x \in A_x$, $y \in A_y$:

$$H(X, Y) = - \sum_{x \in A_x} \sum_{y \in A_y} p_{X,Y}(x, y) \log_2 p_{X,Y}(x, y) \quad (6)$$

It is interesting to mention that, since entropy is a measure of uncertainty, the triangle inequality $H(X, Y) \leq H(X) + H(Y)$ "points out" that having two correlated views reduces uncertainty with direct consequences in haptic data coding. Similarly the inequality of the conditional entropy between two random variables $H(X|Y) \leq H(Y)$ indicates that the LoD rendering is heavily influenced by the by the overall rendering information.

Similarly mutual information between the two random variables X and Y is defined as follows:

$$I(X, Y) = \sum_{x \in A_x} \sum_{y \in A_y} p_{X,Y}(x, y) \log_2 \frac{p_{X,Y}(x, y)}{p_X(x) p_Y(y)} \quad (7)$$

or

$$I(X, Y) = H(X) + H(Y) - H(X, Y) \quad (8)$$

The mutual information $I(X; Y)$, that specifies the amount of information provided by Y about X , is symmetric and non negative. Mutual information can be thus very easily used as a high level information theoretic similarity measure between haptic media.

In particular similarity can be estimated on a region basis e.g. LoD representation, so as to be able to estimate entropies. There is, however, one important drawback to mutual information as a way of comparing vector fields; it fails to take the topology into account since it is calculated only over vector/force values, and not voxel positions. This limitation can be however overcome by increasing the dimensionality of the problem and including geometry information.

4 DEVICE AND PERCEPTUAL HAPTIC FILTERS

Haptic media are assumed to be defined within the input haptic space W that refers to a virtual environment augmented with haptic information. W actually defines

Property	Phantom Omni	Phantom Desktop
Workspace	160Wx120Hx70D mm	160Wx120Hx120D mm
Position resolution	0.055 mm	0.023 mm
Continuous Force	0.88 N	1.75 N

Table 1: Haptic devices properties: “Workspace corresponds to the size of the volume where the haptic probe can move, “positional resolution” to the sensing resolution in mms and continuous force to the maximum force that can be continuously applied on the device.

a mapping of the \mathfrak{R}^3 Euclidean space into the \mathfrak{R}^3 force field that refers to the force exerted to a point object lying on the specific point in space. It should be emphasized that in the general case the haptic space evolves over time and therefore the static W representation becomes a timeseries W_t .

Device Haptic Filters: It is apparent that since haptic media are defined in \mathfrak{R}^3 compared to \mathfrak{R}^2 of the visual media, their storage complexity becomes in the general case $O(n^3)$. However, unfortunately haptic displays have not evolved similarly to visual displays and therefore the spatial resolution they can provide to the user is extremely low with respect to the sensing potential of humans. This is reflected to a very low workspace mapping ratio (equation 4) that can be rendered using typical devices. In information theoretic terms we call this effect as device filter that limits the amount of information that can be perceived. The device filter actually performs a quantization of both the input space \mathfrak{R}^3 in terms of the device spatial resolution and the \mathfrak{R}^3 output space in terms of the force magnitude that can be exerted. Considering now without loss of generality, the case of input space quantization, the device filter is a workspace mapping C_D as defined in Section 3.1, where the workspace refers to the specific haptic device workspace characteristics.

Now let us consider the case of two specific popular haptic devices, namely the Sensable Phantom Omni and Phantom Desktop. Table 1 presents some of their basic properties of interest in our analysis.

Now let us emphasize on the position resolution property, where the Phantom Desktop exhibits almost double the resolution of the Omni. Consider now a static haptic signal S_W defined in a space $Q \in R^3$ and sampled with a resolution of $0.01mm$. Assuming now that the size of Q is equal to the workspaces of the haptic devices, it becomes evident that the haptic information loss for the case of the Omni HIL_O will be always higher or equal to the one of the Desktop HIL_D . In other words, when using the Phantom Omni a lower amount of information is perceived by the end user, since the device filter of the Omni C_O filters out more information with respect to the Desktop filter C_D . Therefore, following the concept of the “alternative soft HR system”, illustrated in Figure 2d, a device dependent en-

coding scheme would transform (e.g. resampling, low-pass filtering, etc.) the initial signal S_W into a new one S_T that produces minimal HIL . This transformation can be formulated as an optimization problem as follows:

$$\mathbf{t} = \underset{\mathbf{t}}{\arg \min} (HIL(S_T, C_D))$$

where \mathbf{t} is the state vector of the transformation of S_W into S_T .

Now, for the case of the “hard HR system”, illustrated in Figure 2b, the transmitter should encode the signal as is, unless information is available about the supported haptic devices. In the latter case the transformation should be applied only for the device of higher resolution and therefore the zero HIL would be valid only for this specific device

Perceptual Haptic Filters:

Besides device display capacity, the perceptual capabilities of the human with respect to temporal resolution have specific limitations [Hirche05], [Hinterseer08], [Kuschel09] that can be used in order to compress haptic media.

Some approaches presented in the literature focusing on telepresence and teleaction [Hinterseer08] tackle the problem of perceptual coding, for the specific case of point-based interaction, trying to define masking thresholds of force differences between consecutive frames beyond which a difference in the force fed back to the user cannot be perceived.

The complexity of this problem for the general case of haptic media and general multi DoF haptic devices is very high, since it does not only depend on the temporal relation between two successive force stimuli but also on the body part that they are applied, the spatial proximity of concurrent stimuli etc. Since, a detailed analysis is out of the scope of present paper, without loss of generality, let us consider the “deadband” approach described in [Hinterseer08].

In particular, in [Hinterseer08] a perceptual mechanism for haptic data compression is proposed based on Weber’s Law and the “Just Noticeable Differences” (JND) principle. At a glance, for specific timeseries of multi-dimensional in the general case haptic data, at a specific time instance the respective haptic sample is transmitted

only if it will be perceived by the user, i.e. if it lies outside the perceptual mask defined by the previously transmitted samples.

This procedure can be seen as a *perceptual filter* that is applied on the input data stream modifying its entropy. Then all quantities described in Section 3 are applicable on the perceptually filtered input. Now the design problem lies on the definition of a perceptual, multivariate in the general case, filter that takes into account the potential haptic rendering schemes, in terms of number of degrees of freedom, perceptual correlation between them, etc.

5 APPLICATIONS AND DISCUSSION

The potential applications of an information theoretic framework of haptic rendering are numerous. First of all, the proposed representation is general, holistic and can accommodate interaction-based rendering as a special case. Moreover, typical multimedia operations including off-line processing, like haptic editing, optimization and indexing become possible. Additionally, in the previous sections more potential applications were outlined including LoD haptic rendering and compression, similarity estimation and design of device and perceptual haptic filters necessary for haptic media coding. Other interesting applications include optimal adaptive workspace partitioning in disjoint subspaces so as to minimize information loss taking into account the rendering device or even correlation analysis between haptic media and visual media, etc.

However, concerning limitations, it should be emphasized that the dimensionality of haptic perception is in the best case much higher with respect to visual perception as very well described in [Hayward11] and in the worst case infinite. Therefore, the transition of interaction-based rendering to open-loop stand-alone streaming haptic media, may introduce a significant increment in dimensionality resulting also in vast amount of haptic information to be transmitted and rendered if not dealt with explicitly. Typical solutions of the computer graphics research community could be employed, like space partitioning and culling. However, to the author's view and as a future research direction, a more fundamental theoretical treatment of dimensionality reduction is necessary so as to result in a complete and tractable information theoretic framework of haptic media. Finally, the proposed framework has to prove its applicability in several applications scenarios, which is a challenging direction for future work.

6 CONCLUSIONS

In this paper an information theoretic view of haptic rendering is presented. Following the major principles of audiovisual communications, several information theoretic quantities are instantiated for the case of

haptic rendering and their potential use in the design of haptic media filters and challenging applications is outlined. It should be emphasized that the proposed framework aims to highlight a different way on how haptic rendering could be potentially dealt with, targeting at haptic media that can be processed, edited, indexed as a stand alone entity and not only as a result of the interaction between a user and the media space.

7 REFERENCES

- [Barbic09] J. Barbic and D. James, "Six-dof haptic rendering of contact between geometrically complex reduced deformable models: Haptic demo," in *In Proc. of Eurohaptics*, March 2009, pp. 393–394.
- [Barlit07] A. Barlit and M. Harders, "Gpu-based distance map calculation for vector field haptic rendering," in *Eurohaptics*, March 2007, pp. 589–590.
- [Borst05] C. Borst, "Predictive coding for efficient host-device communication in a pneumatic force-feedback display," in *Proc. First Joint Eurohaptics Conf. Symp. Haptic Interfaces for Virtual Environ. Teleoperator Syst., Pisa, Italy*, Mar. 2005, pp. 596–599.
- [Cha09] J. Cha, Y.-S. Ho, Y. Kim, and J. Ryu, "A framework for haptic broadcasting," *IEEE Multimedia*, vol. 16, no. 3, pp. 16–27, July, 2009.
- [Chinchor10] N. Chinchor, M. Christel, and W. Ribarsky, "Multimedia analytics," *IEEE Computer Graphics and Applications*, vol. 30, no. 5, pp. 18–19, September 2010.
- [Cirio11] G. Cirio, M. Marchal, A. L. Gentil, and A. Lecuyer, "Six-dof haptic rendering of contact between geometrically complex reduced deformable models: Haptic demo," in *IEEE Virtual Reality Conference*, March 2011, pp. 123–126.
- [Hayward11] V. Hayward, "Is there a plenhaptic function?" *Phil. Trans. R. Soc. B*, no. 366, pp. 3115–3122, 2011.
- [Hikichi01] K. Hikichi, H. Morino, I. Fukuda, S. Matsumoto, Y. Yasuda, M. I. I. Arimoto, and K. Sezaki, "Architecture of haptics communication system for adaptation to network environments," in *Proc. IEEE Int. Conf. Multimed. Expo., Tokyo, Japan*, Aug. 2001, pp. 744–747.
- [Hinterseer08] P. Hinterseer, S. Hirche, S. Chaudhuri, E. Steinbach, and M. Buss, "Perception-based data reduction and transmission of haptic data in telepresence and teleaction systems," *IEEE Transactions on Signal Processing*, vol. 56, no. 2, pp. 588–597, February, 2008.
- [Hirche05] S. Hirche, A. Bauer, and M. Buss, "Transparency of haptic telepresence systems with con-

- stant time delay,” in *in Proc. IEEE Int. Conf. Control Appl., Toronto, Canada, 2005*, pp. 328–333.
- [Kaklanis09] N. Kaklanis, D. Tzovaras and K. Moustakas, “Haptic navigation in the World Wide Web,” in *HCI International 2009*, pp. 707–715, San Diego, July 2009.
- [Kaklanis10] N. Kaklanis, K. Votis, K. Moustakas and D. Tzovaras, “3D HapticWebBrowser: Towards Universal Web Navigation for the Visually Impaired,” *7th International Conference on Web Accessibility, W4A 2010*, Raleigh, USA, April 2010.
- [Kostopoulos07] K. Kostopoulos, K. Moustakas, D. Tzovaras, G. Nikolakis, C. Thillou, and B. Gosselin, “Haptic access to conventional 2d maps for the visually impaired,” *Springer International Journal on Multimodal User Interfaces*, vol. 1, no. 2, pp. 13–19, July 2007.
- [Kron04] A. Kron, G. Schmidt, B. Petzold, M. F. Zah, P. Hinterseer, and E. Steinbach, “Disposal of explosive ordnances by use of a bimanual haptic telepresence system,” in *in Proc. IEEE Int. Conf. Robot. Autom., New Orleans, LA, Apr. 2004*, pp. 1968–1973.
- [Kuschel09] M. Kuschel, P. Cremer, and M. Buss, “Passive haptic data-compression methods with perceptual coding for bilateral presence systems,” *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 39, no. 6, pp. 1142–1151, November, 2009.
- [Lin08] M. Lin and M. Otaduy, “Haptic rendering: Foundations, algorithms and applications,” 2008, p. A.K.Peters publishing.
- [Laycock07] S. Laycock and A. Day, “A survey of haptic rendering techniques,” *Computer Graphics Forum*, vol. 26, no. 1, pp. 50–65, January 2007.
- [MacLean08] D. Ternes and K. MacLean, “Designing large sets of haptic icons with rhythm,” *Springer LNCS, Haptics: Perception, Devices and Scenarios*, vol. 5024, pp. 199–208, 2008.
- [Moustakas06] K. Moustakas, D. Tzovaras, S. Carbini, O. Bernier, J.E. Viallet, S. Raidt, M. Mancas, ..M. Dimiccoli, E. Yagci, S. Balci, E.I. Leon and M.G. Strintzis “MASTERPIECE: Physical Interaction and 3D content-based search in VR Applications,” *IEEE Multimedia*, vol. 13, no. 3, pp. 92–100, July 2006.
- [Moustakas07] K. Moustakas, G. Nikolakis, K. Kostopoulos, D. Tzovaras, and M. Strintzis, “Haptic rendering of visual data for the visually impaired,” *IEEE Multimedia*, vol. 14, no. 1, pp. 62–72, January 2007.
- [Moustakas07b] K. Moustakas, D. Tzovaras, and M. Strintzis, “Sq-map: Efficient layered collision detection and haptic rendering,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 1, pp. 80–93, January 2007.
- [Nikolakis06] G. Nikolakis, D. Koutsonanos, P. Daras, K. Moustakas, D. Tzovaras, and M. Strintzis, “Haptic interaction in medical virtual environments,” *Wiley Encyclopedia of Biomedical Engineering, 6- Volume Set, Metin Akay (Editor)*, ISBN: 0-471-24967-X, June 2006.
- [Ortega02] A. Ortega and Y. Liu, “Lossy compression of haptic data,” in *in Touch in Virtual Environments: Haptics and the Design of Interactive Systems. Englewood Cliffs, NJ: Prentice-Hall, ch. 6*, 2002, pp. 119–136.
- [Ou02] E. Ou and C. Basdogan, “Network considerations for a dynamic shared haptic environment,” in *in Proc. Nat. Conf. Undergrad. Res. Whitewater, WI, April, 2002*.
- [Palmerius08] K. Palmerius, M. Cooper, and A. Ynnerman, “Haptic rendering of dynamic volumetric data,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 263–276, April 2008.
- [Petersik01] A. Petersik, B. Pflesser, U. Tiede, and K. Hohne, “Haptic rendering of volumetric anatomic models at sub-voxel resolution,” in *In Proc. of Eurohaptics, Birmingham, UK, 2001*, pp. 182–184.
- [Shannon48] C. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.
- [Souayed03] R. T. Souayed, D. Gaiti, G. Pujolle, W. Yu, Q. Gu, and A. Marshall, “Haptic virtual environment performance over ip networks: A case study,” in *in IEEE Symp. Distrib. Simul. Real-Time Appl., Delft, Netherlands, Oct. 2003*, pp. 181–189.
- [Srinivasan97] M. Srinivasan and C. Basdogan, “Haptics in virtual environments: Taxonomy, research status and challenges,” *Computers and Graphics*, pp. 393–404, 1997.
- [Torre06] G. Robles-De-La-Torre, “The importance of the sense of touch in virtual and real environments,” *IEEE Multimedia*, vol. 13, no. 3, pp. 24–30, July 2006.
- [Vittorias09] I. Vittorial, J. Kammerl, S. Hirche, and E. Steinbach, “Perceptual coding of haptic data in time-delayed teleoperation,” in *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Worldhaptics 2009*, Mar. 2009, pp. 208–213.
- [Zadeh08] M. Zadeh, D. Wang, and E. Kubica, “Perception-based lossy haptic compression considerations for velocity-based interactions,” *Multimedia Systems*, vol. 13, pp. 275–282, 2008.

Deformation by discrete elastica

Flavio Bertini

Department of Computer
Science and Engineering
University of Bologna
Mura Anteo Zamboni 7
40126 Bologna, Italy
fbertini@cs.unibo.it

Serena Morigi

Department of
Mathematics
University of Bologna
P.zza Porta San Donato 5
40126 Bologna, Italy
serena.morigi@unibo.it

ABSTRACT

Modeling a physically plausible deformation of a flexible object from its naturally planar (curved) state is a fundamental and challenging topic in digital surface processing with applications to computer animation and game design. We propose a new variational model for detail preserving surface-based deformation of a body based on total curvature energy. We demonstrate the efficacy of the model with several examples which enhance the realism of the deformation.

Keywords

Mesh deformation, total curvature, surface processing, elasticity.

1 INTRODUCTION

The great interest in computer animation and game design has led over time to the development of a large amount of deformation and editing methodologies for discrete models [12]. Surface based-deformation methods have been recently widely investigated and represent an emerging alternative both to *rigging* (i.e., adding a skeleton to control and animate a mesh) which is the classical way in the graphics industry to efficiently design poses and deformation, and *caging* which uses control structure like lattices to immerse the object and propagate the deformation.

In this paper we propose a new variational surface-based deformation formulation which improves traditional Laplacian surface editing by using the total curvature as a better aesthetic measure for deformation of elastic bodies.

The basic idea of the variational design approach is to measure the quality of a surface in terms of a certain curvature-based energy. In general, a curvature energy may be expressed in terms of principal curvatures of a surface: mean ($H = k_1 + k_2$), Gaussian ($K = k_1 k_2$), and total ($k_1^2 + k_2^2$) curvature. The relationship between curvature and bending energy first appeared in an explicit

form in Euler's elastica, curves minimizing the integral of squared curvature.

Let us denote by \mathcal{M} a two-manifold surface, parameterized by a function $X : \Omega \subset \mathbb{R}^2 \rightarrow \mathcal{M} \subset \mathbb{R}^3$, where Ω is an open reference domain.

Thin flexible structures are governed by a surface bending energy of the form

$$E(\mathcal{M}) := \int_{\mathcal{M}} \alpha + \beta(H - H_0)^2 - \gamma K d\mathcal{M}, \quad (1)$$

the so-called Canham-Helfrich model [14], where H_0 denotes the spontaneous curvature which plays an important role in thin-shell. For $\alpha = H_0 = 0$ and $\beta = 1, \gamma = 2$, the Canham-Helfrich model reduces to the **total curvature energy**

$$E_T(\mathcal{M}) := \int_{\mathcal{M}} (H^2 - 2K) d\mathcal{M} = \int_{\mathcal{M}} (k_1^2 + k_2^2) d\mathcal{M}, \quad (2)$$

which approximates the bending energy of a thin plate manifold. In (2) the principal curvatures k_1 and k_2 depend non-linearly on the surface \mathcal{M} . Let us call the surfaces minimizing (1) elastica surfaces because they generalize the famous Euler's elastica curves. This energy is invariant under rigid motions and uniform scaling of the surface \mathcal{M} , that is, it is invariant under Möbius transformations.

In (2) the first term ($E_B(\mathcal{M}) := \int_{\mathcal{M}} H^2$) represents the well-known Willmore energy [33], successfully used in surface fairing and restorations [9].

According to the Gauss-Bonnet Theorem from differential geometry [11], the integral over a disk region

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

$\mathcal{M} \subset \widetilde{\mathcal{M}}$ can be expressed as an arclength integral over the boundary $\partial\mathcal{M}$ as follows

$$\int_{\mathcal{M}} K d\mathcal{M} = 2\pi - \oint_{\partial\mathcal{M}} k_g ds, \quad (3)$$

where k_g is the geodesic curvature of the boundary curve $\partial\mathcal{M}$. This implies that the Gaussian curvature of \mathcal{M} depends only on a collar neighborhood of $\partial\mathcal{M}$: if we make any modification to \mathcal{M} away from the boundary, the total curvature is unchanged (as long as \mathcal{M} remains topologically a disk).

That is, the second term in (2) remains constant on surfaces with fixed boundaries and fixed normals on the boundary. This is the main reason why the surface-based deformation strategies proposed so far limit the deformation model to the Willmore energy.

However, when the deformation acts on some interior region of a closed compact manifold $\widetilde{\mathcal{M}}$, then \mathcal{M} has fixed boundaries but changing normals, and for an open manifold the deformation region can even have free boundaries and normals. Moreover, a deformation can easily change even the topology of the surface, thus changing the Gaussian curvature of it. These observations are the major motivations for the work presented here.

Therefore, the second term in (2) cannot be neglected in the energy optimization process, and only including it in the deformation process will allow us to handle a wider class of free-form deformations.

Moreover, both the energy $E_T(\mathcal{M})$ and $E_B(\mathcal{M})$ are invariants under all Möbius transformations (in particular, rigid motions and uniform scaling of the surfaces). In extending the energies from smooth surfaces to the discrete case (polyhedral surfaces) we will require this property to remain true.

In this paper we are interested in modeling a physically plausible deformation of a flexible object from its naturally planar (curved) state to an equilibrium configuration due to external localized forces interactively applied by the user (by imposing geometrical constraints). As energy of deformation associated with the elastically deformable object we consider the total curvature functional for the justification given above. This energy is employed to define the internal elastic force of the object, expressed as δE , a first variational derivative of the potential energy of deformation. To deal effectively with boundaries we introduce appropriate boundary conditions which guarantee, when it is possible, to satisfy G^1 continuity conditions at the boundary of the domain.

The rest of the paper is organized as follows: in section 2, some related work on mesh deformation is presented and in section 3, we discuss some potential energies of deformations and a new variational deformation model

which includes the effect of the total curvature. Linear and nonlinear deformation constraints are described in section 4. In section 5, the details on the discretization are given, and we present the deformation algorithm based on the iterative alternating strategy. The effect of the total curvature energy and other examples are discussed in section 6. Some limits and concluding remarks are made in section 7.

2 RELATED WORKS

Deformable curve, surface, and solid models gained popularity in computer vision and computer graphics after they were proposed in the mid 1980s by Terzopoulos et al. [30]. Terzopoulos introduced the theory of continuous (multidimensional) deformable models in a Lagrangian dynamics setting, based on deformation energies in the form of (controlled-continuity) generalized splines [31]. Since then, many results have been presented in that direction. Nowadays, most existing shape deformation approaches can be classified as *space deformation* and *surface deformation* according to the way in which they act on the object to be deformed [4]. Space deformation techniques modify objects by deforming their embedded space, while surface-based methods define the deformation directly on the surface of the object.

All space deformation methods use some form of control structure like a lattice to immerse the object then every structure deformation is propagated to the object itself. The robustness and the efficiency of these methods are strongly affected by the control structure complexity even if they are less affected by the complexity or the triangles' quality of the original surface [27]. In *Free-Form Space Deformation* points of the object are expressed as a linear combination of the structure control points and blended with some different bases functions: Bézier [23], B-spline [13] [17], T-splines [27] and many others. The unnaturalness of the deform through a control structure has led to the development of the *Direct Manipulation FFD* [15]: the user directly moves the object points and the system computes the control point displacements. The *Radial Basis Functions*-based approach proposed in [3] is equally innovative, it improves upon FFD and DMFFD due to its handle point nature and the deformation function physically used.

Nonlinear methods are presented in [28], where an energy functional optimizes the local deformation gradients, and in [6], where the object is voxelized and the deformation is driven by a nonlinear elastic energy.

There are a plenty of different surface deformation methods, well summarized in the detailed survey [7]. The *Transformation Propagation* linearly propagates the deformation within a region and the main challenge is how to define the propagation function (e.g. using

geodesic distances [2] or Euclidean distances [22]). The *Shell-Based Deformation* techniques minimize two physically-inspired deformation energies: stretching and bending [32]. The *Multi-scale Deformation* [8] mainly decomposes the object into two frequency bands: high frequency for details and low frequency for global shape. Two detail-preserving techniques have been proposed as surface deformation methods based on differential coordinates: the *Gradient-Based Deformation* [35] and the *Laplacian-Based Deformation* [26],[18]. The former uses the original surface gradients as target in the least-squares sense for the deformed surface, the latter is similar but it uses the Laplacian operator on vertices.

The linear methodologies can be solved very efficiently but can lead to counterintuitive results for large-scale deformations, they have obvious limitations and in some circumstances they even fail. Nonlinear deformation techniques overcome these limitations, but they require more complex numerical schemes [4].

Typically, some constraints are added to improve quality results: *Pyramid coordinates* [24], handle-aware isoline technique [1], volumetric graph Laplacian [16], skeleton-based inverse kinematics [25] and shell-based minimization coupled by a nonlinear elastic energy [5].

Another important issue is the deformation metaphor which concerns the manner in which the user defines the deformation: by *handle point deformation* the user moves some “handle” points of the object, by *curve-based deformation* the user imposes deformation by sketching curves and by *control point deformation* the user manages the object in an indirect way [12].

3 ENERGIES OF DEFORMATION

The deformation of a nonrigid body is a change in shape or size of an object due to applied forces: pulling or pushing (compressive) forces, shear, bending or torsion (twisting). A deformation is termed *elastic* if the undeformed or reference shape restores itself completely, upon removal of all external forces, *inelastic* otherwise. A pioneer work in modeling inelastic deformations simulating behaviors such as viscoelasticity, plasticity, and fracture, has been proposed in [30] for use in computer graphics animation of nonrigid objects. More advanced physically-based PDE models are considered in [21].

In this work, we focus on elastic deformations of a non-rigid model during the deformation phase neglecting the phase responsible for recovering the reference shape.

Let \mathcal{M} be a fixed reference surface, parameterized by a function $X : \Omega \subset \mathbb{R}^2 \rightarrow \mathcal{M} \subset \mathbb{R}^3$. A deformation is a function d that maps \mathcal{M} to a certain deformed model \mathcal{M}' , by adding to each point $X(u, v) \in \mathcal{M}$ a displacement vector $d(u, v)$, such that $\mathcal{M}' = X'(\Omega)$, $X' = X + d$.

A reasonable approximation for elastic thin-shell energy which measures stretching and bending is the following (see [4] for details)

$$\int_{\Omega} k_s \|I' - I\|^2 + k_b \|II' - II\|^2 dudv, \quad (4)$$

where I (I') and II (II') represent the first and second fundamental forms for \mathcal{M} (\mathcal{M}'), k_s and k_b weight the matrix norms and determine resistance to stretching and bending, respectively. The matrix norms in this function make it highly nonlinear, leading to a difficult nonlinear optimization problem. It is therefore common to simplify (linearize) this objective function by replacing the change of the first and second fundamental forms by the first-order and second-order partial derivatives of the displacement function d [7].

The (simplified) **thin plate energy** is given by:

$$E_B(d) = \frac{1}{2} \int_{\Omega} k_b (\|d_{uu}\|^2 + 2\|d_{uv}\|^2 + \|d_{vv}\|^2) dudv. \quad (5)$$

The stretching or **membrane energy** is defined by the functional

$$E_M(d) = \frac{1}{2} \int_{\Omega} k_s (\|d_u\|^2 + \|d_v\|^2) dudv. \quad (6)$$

Deformations of non-rigid surfaces (so called *thin-shells*) require both energies, while for a deformable solid only the local stretching within the object is considered.

In order to keep the parametrization of the surface \mathcal{M} as close to isometric as possible, Ω is typically chosen to be equal to the initial surface \mathcal{M} , such that $d : \mathcal{M} \rightarrow \mathbb{R}^3$ is defined on the manifold \mathcal{M} itself. As a consequence, the Laplace operator Δ w.r.t. the parametrization X turns into the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$ w.r.t. the manifold \mathcal{M} . Therefore when the parametrization is isometric,

$$E_B(d) \simeq \frac{1}{2} \int_{\mathcal{M}} k_b \|\Delta_{\mathcal{M}} d\|^2 d\mathcal{M}, \quad (7)$$

and

$$E_M(d) \simeq \frac{1}{2} \int_{\mathcal{M}} k_s \|\nabla_{\mathcal{M}} d\|^2 d\mathcal{M}. \quad (8)$$

The minimization of these functionals, performed efficiently by applying variational calculus, yields to their Euler-Lagrange equations which are

$$\Delta_{\mathcal{M}}^2 d = 0, \quad (9)$$

for (7), and

$$-\Delta_{\mathcal{M}} d = 0, \quad (10)$$

for (8), subject to suitable boundary constraints. We denote by E_C the variational deformation proposed in [7]

where stretching and bending are combined together, given by

$$-k_s \Delta_{\mathcal{M}} d + k_b \Delta_{\mathcal{M}}^2 d = 0. \quad (11)$$

We observe that in surface smoothing similar functionals are applied to X itself instead of their displacements [10].

Note that the linearization in (7) and (8) causes artifacts for large deformations, as we will verify in the result section 6.

We propose instead to minimize the total curvature energy (2), which leads to the Euler-Lagrange equation

$$\Delta_{\mathcal{M}} H(d) - 2H(d)(H^2(d) - K(d)) = 0, \quad (12)$$

subject to natural boundary conditions. Equation (12) is a fourth-order partial differential equation, (the term $\Delta_{\mathcal{M}} H(X)$ involves fourth-order surface derivatives) satisfied for an elastica surface. To be well posed it requires two independent boundary conditions. By natural boundary conditions we mean that no continuity conditions are specified at the boundary points, but the continuity is implied by the "outer" part incident to the boundary of \mathcal{M} . The elastica flow has been proposed for surface fairing and repairing in [34]. Note that $\sqrt{H^2 - K}$ is the half-difference of the principal curvatures and also in the discrete setting the property $H^2 - K \geq 0$ has to be guaranteed.

In a modeling application, one can be interested either to a dynamic time dependent simulation, or directly to solve the rest state of the deformation process. For the former, one typically deals with the associated geometric flow $\partial \mathcal{M} / \partial t = -\nabla(E(d))$ by the steepest descent method. The latter means solving the Euler-Lagrange formulation subject to user-defined boundary constraints, which will be discussed in section 4. This typically means to fix certain surface regions $\mathcal{F} \subset \mathcal{M}$, and to define displacements for the so-called handle (target) regions $\mathcal{H} \subset \mathcal{M}$. In an interactive application \mathcal{M} has to be recomputed by solving the PDE each time the user manipulates the boundary constraints, for instance by moving the handle region \mathcal{H} .

Considering the deformation energy together with linear or nonlinear constraints represented by a generic $\Phi(\cdot)$, the constrained deformation can be formulated by

$$\min_{X'} E(X' - X) \quad \text{subject to} \quad \Phi(X'). \quad (13)$$

4 LINEAR AND NONLINEAR DEFORMATION CONSTRAINTS

First we consider positional constraints that can be either incorporated as **hard** or **soft** constraints. The hard positional constraints are preferred in classical editing tools where the exact position should be achieved,

while in a sketch-based system soft constraints are actually advantageous, since they allow the user to place imprecise locations to hint the desired shape, without specifying it exactly.

Representing the coordinate map X by real-valued functions (x, y, z) , defined on \mathcal{M} , the prescribed target positions C are imposed by the following constraints:

$$\frac{1}{2} \int_{\mathcal{M}} (X' - C)^2 d\mathcal{M}. \quad (14)$$

Intuitive, detail-preserving constraints can be obtained by preserving local differential properties under deformation.

Let $\delta = \Delta_{\mathcal{M}}(X)$, the surface deformation is obtained by

$$\frac{1}{2} \int_{\mathcal{M}} (\Delta_{\mathcal{M}} X' - \delta)^2 d\mathcal{M}. \quad (15)$$

Eq. (15) forces the new position to resemble its undeformed Laplacian as closely as possible, that is, in view of the fact that $\Delta_{\mathcal{M}} X = -H\mathbf{n}$, with \mathbf{n} outward surface normals, it preserves the local curvatures of the undeformed surface. Laplacian deformation methods are mainly based on the minimization of (15), see [26],[7].

This deformation constraint tries to preserve the orientation of the normals w.r.t. the global coordinate system, whereas in reality they should rotate with the deformed surface. This technique as well as the gradient-based deformation [35] fail to yield intuitive results for translational deformations as it clearly shown in section 6, since a translation does not cause a change in surface gradients, or normal vectors. This *translation insensitivity* is an inherent limitation of most approaches based on differential coordinates.

A correct deformation should retain the local surface features, that is their relative orientation and possibly their size. Therefore, several variants to the linear model (15) have been proposed for finding local rotations of the geometric details: some require additional input of the rotation of the handle, others use implicit optimizations, and multiresolution approaches estimate the local transformation from the deformation of the base surface.

Introducing the local transformations T , which can be restricted to rotation and isotropic scaling, the differential representations δ of the rest mesh X are transformed into the deformed pose: $\hat{\delta} = T\delta$. The deformed positions X' are then obtained by replacing (15) with the following nonlinear term

$$\frac{1}{2} \int_{\mathcal{M}} (\Delta_{\mathcal{M}} X' - \hat{\delta}(X'))^2 d\mathcal{M}. \quad (16)$$

The term $\hat{\delta}(X')$ is a nonlinear function because it includes the effects of local rotations, thus (16) leads to a nonlinear least-squares problem, while (15) leads to a linear least-squares problem. We impose the nonlinear constraints on the set $\mathcal{M} \setminus (\mathcal{F} \cup \mathcal{H})$.

5 MODEL DISCRETIZATION

We are interested in evaluating the constrained variational problem involving the curvature energy (2) for discrete surfaces, i.e., triangular meshes M which represent a piecewise-linear approximation of the smooth surfaces \mathcal{M} . Let M be defined by a set T of triangles $T_i, i = 1, \dots, N_t$, that cover M , and a set X of vertices $X_i, i = 1, \dots, N_v$, where $X_i \in \mathbb{R}^3$ is the i th vertex, $X_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, with associated normal vector \mathbf{n}_i .

Let us briefly introduce the key ingredients for the discretization on M of the proposed variational deformation formulation. Since for a smooth surface $\Delta_{\mathcal{M}} = 2H\mathbf{n}$, see [11], a discrete approximation of the mean curvature vector $H_i\mathbf{n}_i$ associated to the vertex X_i can be derived using the following discrete form

$$H_i\mathbf{n}_i = L(X_i) = \frac{1}{2A_i} \sum_{j \in N(i)} w_{ij}(X_j - X_i), \quad (17)$$

where L represents the discretization of the local Laplace-Beltrami operator $\Delta_{\mathcal{M}}$ on M , $N(i)$ is the set of 1-ring neighbor vertices of vertex X_i , A_i is the Voronoi area surrounding X_i , and the weights w_{ij} are positive numbers which satisfy the normalization condition $\sum_{j \in N(i)} w_{ij} = 1$. Different geometric discretizations of the Laplacian can be obtained for different choices of the weights in (17), the most common and used in our discretization, introduced by Meyer et al. in [19], is

$$w_{ij} = (\cot \alpha_{ij} + \cot \beta_{ij}), \quad (18)$$

where α_{ij} and β_{ij} are the two angles opposite to the edge in the two triangles sharing the edge (X_j, X_i) .

The notion of Gaussian curvature extends to such discrete surface M and is supported on the vertices $X_i \in X$. In fact, to keep the Gauss-Bonnet theorem true, we must take

$$\int_{\mathcal{M}} K d\mathcal{M} := \sum_i K_i, \quad K_i = \frac{1}{A_i} (2\pi - \sum_{j \in N(i)} \theta_j), \quad (19)$$

where θ_i are the incident internal angles at X_i . The simple formula (19) is a standard for triangular meshes [29].

Considering that the displacement vector is $d = X' - X$, then the deformation energy models on the surface represented by the mesh M can be discretized as follows:

E_M (10)	$Ld = 0$	$LX' = LX$
E_B (9)	$L^2d = 0$	$L^T LX' = L^T LX$
E_C (11)	$k_s Ld + k_b L^2d = 0$	$(k_s L + k_b L^2)X' = (k_s L + k_b L^2)X$
E_T (12)	$L^2d - 2LGd = 0$ $G = H^2 - K$	$(L^2 - 2LG)X' = (L^2 - 2LG)X$

Each of these models leads to a generic linear system $AX = b$ with a sparse $N_v \times N_v$ coefficient matrix. Note

that, in general, the evaluation of $\Delta_{\mathcal{M}}H$ at X_i (for X_i either being an inner, that is $X_i \in X \setminus (\mathcal{F} \cup \mathcal{H})$, or an outer vertex, that is $X_i \in \partial\mathcal{M}$) involves 2-ring neighbor vertices of X_i . Some of them may be inner vertices, and the remaining are outer vertices. The inner vertices are treated as unknowns in the discretized equations and the outers are incorporated into the right-hand side.

A way to enforce *soft positional constraints*, is to incorporate them in a penalty formulation of the discrete energy functional

$$\min_{X'} E(X' - X) + \frac{\lambda}{2} \|X' - C\|^2 \quad (20)$$

where $\lambda > 0 \in \mathbb{R}^n$ is the penalty coefficient, and C is the vector of prescribed vertex positions.

In order to approach the interpolation of the constraints C , the parameter λ has to be chosen sufficiently large. However, the condition number of the matrix grows with λ , then a higher weight can cause numerical problems.

Considering the discretization of (13) with total curvature energy (2), positional constraints (20) and detail-preserving constraints (16), and using the fact that $\hat{\delta}(X')$ are unknowns in the deformation process, we propose the following energy functional minimization to solve the mesh deformation problem:

$$\min_{X', \hat{\delta}} \mathcal{L}(X', \hat{\delta}), \quad \mathcal{L}(X', \hat{\delta}) := E(X' - X) + \frac{\lambda_1}{2} \|X' - C\|^2 + \frac{\lambda_2}{2} \|LX' - \hat{\delta}\|^2. \quad (21)$$

The minimum of (21) can be determined by the alternating minimization procedure, namely, for $k = 0, 1, \dots$, we solve successively

$$\begin{aligned} \hat{\delta}^{(k+1)} &= \operatorname{argmin}_{\hat{\delta}} \mathcal{L}(X'^{(k)}, \hat{\delta}) \\ X'^{(k+1)} &= \operatorname{argmin}_{X'} \mathcal{L}(X', \hat{\delta}^{(k+1)}). \end{aligned} \quad (22)$$

Since $\mathcal{L}(X', \hat{\delta}^{(k+1)})$ is continuous differentiable in X' , the solution $X'^{(k+1)}$ of the second minimization in (22) is obtained by imposing

$$0 = \nabla_{X'} \mathcal{L}(X', \hat{\delta}^{(k+1)}) = (L^2 - 2LG)(X' - X) + \lambda_1(X' - C) + \lambda_2(L^T(LX' - \hat{\delta}^{(k+1)})), \quad (23)$$

where $G = H^2 - K$.

In matrix-vector form, the solution of (23) for the new mesh vertices X' , is given by solving the overdetermined system

$$\begin{bmatrix} (L^2 - 2LG) \\ \mathbf{0} \quad \sqrt{\lambda_1} I_n \\ \mathbf{0} \quad \sqrt{\lambda_2} L^T L \end{bmatrix} X' = \begin{bmatrix} (L^2 - 2LG)X \\ \sqrt{\lambda_1} C \\ \sqrt{\lambda_2} L^T \hat{\delta}^{(k+1)} \end{bmatrix} \quad (24)$$

where the block $A = (L^2 - 2LG)$ represents the internal energy, $I_n \in \mathbb{R}^{n \times n}$ is the identity matrix which requires

a resorting of the rows of L , and $C \in \mathbb{R}^n$ is a vector of elements c_i for each of the n positional constraint. The system has dimension $(N_v + 2n) \times N_v$ and it is full rank, thus it has a unique solution in the least-squares sense. The linear system of equation (24) is solved by the conjugate gradient method where we terminate the iterations as soon as the norm of the residual is less than or equal to 10^{-4} . The use of an iterative solver allows us to avoid storing the large dimension matrices, the only requirement is matrix-vector products.

The use of $L = D\bar{L}$ and $K = D\bar{K}$ with the area-scaling matrix $D_{ii} = 1/A_i$ in (24) allows to extend the invariance under rigid transformations and uniform scaling property of the energy E_T to the discrete setting.

To update $\hat{\delta}^{(k+1)}$, the first minimization in (22) gives $\hat{\delta}^{(k+1)} = LX^{(k)}$. In particular, following [16], at each step we use the two-phase procedure:

- (STEP 1) For each vertex X_i with $N(i)$ neighbors, solve for $\mu^i = (\mu_1^i, \mu_2^i, \dots, \mu_{N(i)}^i)$:

$$\sum_{j \in N(i)} \mu_j^i ((X_j - X_i) \otimes (X_{j-1} - X_i)) = \delta_i, \quad (25)$$

where δ_i are the Laplacian coordinates before deformation, and $(X_j - X_i) \otimes (X_{j-1} - X_i)$ is the normal vector to the triangle X_i, X_j, X_{j-1} on X_i . The overdetermined linear system (25) can be represented in matrix-vector form as

$$A_i \mu^i = \delta_i \quad (26)$$

where A has dimension $3 \times N(i)$ and solved by SVD method.

- (STEP 2) Plug the computed μ_j^i in

$$d_i(X') = \sum_{j \in N(i)} \mu_j^i ((X'_j - X'_i) \otimes (X'_{j-1} - X'_i)), \quad (27)$$

with X' vertices of the deformed mesh M' . Since the μ^i are the same before and after deformation, $d_i(X) = T_i \delta_i$ for local rotations T_i . Finally the Laplacian coordinates are normalized as follows:

$$\hat{\delta}(X_i) = \frac{d_i}{\|d_i\|} \|\delta_i\|. \quad (28)$$

This means that we use the Laplacian coordinates of the previous iterative step as the target direction, while taking the magnitude of the original Laplacian coordinates as the target magnitude. Since the directions to be preserved are those at the rest position, then STEP1 can be done as preliminary step, while STEP2 updates the new Laplacian coordinates at each alternating step k .

In case we want to enforce *hard positional constraints*, that is constrained vertices which lie in the exact prescribed location, we set $\lambda_1 = 0$ in the functional (21),

and solve (24) replacing the matrix block A with a reduced matrix $A_n \in \mathbb{R}^{(N_v-n) \times (N_v-n)}$ as follows. Forcing n constraints leads to the elimination in A of the n rows corresponding to the constrained vertices ($X_i \in \mathcal{F} \cup \mathcal{H}$) and moving the corresponding columns to the right-hand side.

We should remark that the original mesh without the constrained vertices \mathcal{F} and \mathcal{H} is a reduced mesh with, in general, more than one connected components. The associated connectivity matrix should be a reduced rank matrix. However, the A_n matrix is not the connectivity matrix of such a reduced mesh since the elements of A_n are the same of the original connectivity matrix, that is computed on the entire mesh.

The final deformation algorithm simply iterates two simple and efficient steps which are respectively responsible for improving the estimation of the local transformations, and vertex positions. At each iteration, fixing X' , $\hat{\delta}^{(k+1)}$ are updated by using STEP2, then the computed approximations for $\hat{\delta}^{(k+1)}$ are used to solve the linear least-squares problem (24). The algorithm for nonlinear deformation is here summarized:

ALGORITHM

Discrete Elastica Nonlinear Deformation (DEND)

INPUT: undeformed vertex set X ,

OUTPUT: deformed vertex set X'

Set $X^{(0)} = X$, $\hat{\delta}(X^{(0)}) = L(X)$, $k=0$

STEP 1: Compute μ^i , $\forall X_i \in X$ by (25)

Repeat

STEP 2: Compute $\hat{\delta}(X^{(k+1)})$ by (27) and (28)

Solve the linear LS problem (24)

$k=k+1$

until $\|X^{(k)} - X^{(k-1)}\| < 1 \cdot 10^{-3}$

6 DEFORMATION RESULTS

In this section we consider some examples to show how the proposed DEND algorithm performs to deform structured and unstructured polygonal meshes. All the examples have been produced on a standard consumer-level LINUX PC by using the Meshviz software, a GUI application for geometric surface processing developed at the University of Bologna, Italy, based on OpenGL graphics library and C language. The current version of Meshviz offers interactive tools for experimentation with the proposed deformation algorithm but it does not provide any tangential remeshing feature. This would prevent from degenerated meshes which may adversely affect the deformation performance. A preprocessing step of mesh optimization can alleviate this problem [20]. Moreover, the DEND algorithm has no collision detection which would allow for handling collision occurring between deformed parts of a deformable body. Collision detection is a complex constraint which increases considerably the complexity of

the deformation model. The `Meshviz` editing tools are easy and intuitive and allow the user to interactively select by mouse regular/irregular regions \mathcal{F} that he/she wants to keep fixed so as the areas \mathcal{H} that he/she will drag (rotating, translating) to a target position; the positions of the remaining vertices $X \setminus (\mathcal{F} \cup \mathcal{H})$ will be determined by the DEND algorithm. The computational time of the entire process to set up a new pose for an object depends mainly on the dimension of the free vertex set $X \setminus (\mathcal{F} \cup \mathcal{H})$ which affects the solution of (24) in DEND algorithm. Therefore, for medium size objects like those used for the shown examples the deformations are achieved in real-time, while optimizations of the linear solver will be needed to obtain real-time deformations of larger meshes.

We demonstrate that our approach enables to apply small to large deformations on middle-large detailed meshes while keeping the shape of the details in their natural orientation.

Example 1. In this example we compare the surface-based mesh deformation techniques described in section 3. The DEND Algorithm presented in section 5 has been suitably modified to manage the different energy deformations by changing the block matrix A , and hard/soft constraints, by setting $\lambda_1 = 0$ or $\lambda_1 = 1$, respectively.

The deformations are performed on the original undeformed `bar` mesh with 856 vertices (Fig.1(a)) and `cylinder` mesh with 1088 vertices, illustrated in Fig.1 (b). Fig.1, first column, shows the results of 135° twist of the `bar` mesh, and second column reports the 120° bending of the `cylinder` mesh. In particular, we apply (21) with the deformation energies E_B (9) (Fig.1(e)-(f)), E_C (11) (Fig.1(g)-(h)), and E_T (12) (Fig.1(m)-(n)), with non-linear hard constraints ($\lambda_1 = 0, \lambda_2 = 1$). For comparison, we also show in Fig.1(i)-(l) the deformation obtained by the PRIMO system [5], courtesy from the author's web page, which present different, but still physically plausible shape deformations. However, the `cylinder` deformation presents a more concentrate bending in the middle, while using E_T (12) (Fig.1(n)) we get a uniform deformation along the shape. In each figure the red vertices represent the vertices \mathcal{F} that are fixed; the blue vertices are the handle (target) constraints \mathcal{H} ; the green area contains the remaining unconstrained vertices $X \setminus (\mathcal{F} \cup \mathcal{H})$ whose position is computed by the DEND algorithm. Fig.1(c)-(d) show the resulting deformations by using the thin plate energy E_B defined in (9) applied together with linear constraints (15) instead of nonlinear (16). The results clearly show the weaknesses of the linear deformation approach.

In linear theory the behaviour of the deformable model is physically correct only for small displacements (about 10% of the mesh size), it is less realistic for

larger deformations. It is the main disadvantage of linear elasticity. In Fig.1 all the deformations are obtained in a single step. Nevertheless, our deformation procedure by using E_T with non-linear constraints (Fig.1(m)-(n)), provides well-shaped and aesthetically pleasing results. Interactive deformation prevents large deformations, since each step remain reasonably small. Fig.2 illustrates two large deformations obtained by interactively applying 5 steps for a total 300° twisting on the `bar` model (Fig.2(a)), and 3 steps for a 180° bending on the `cylinder` mesh (Fig.2(b)). The other considered energies were not been able to produce similar good results.

Example 2. The example illustrated in Fig.3 shows how to apply a simple deformation to transform a plane into a bumpy plane. From an original plane model, shown in Fig.3 (a), the `bumpy plane` (2115 vertices) in Fig.3 (b) is achieved by anchoring the red vertices and translating the blue vertices. The `bumpy plane` is then deformed by bending the two sides of the mesh using the DEND Algorithm with E_T .

Example 3. In this example we show the flexibility of our variational deformation model implemented with DEND Algorithm on a range of examples, including both open and closed surfaces representing elastically deformable models.

Fig.4 shows the deformations of a thin plate model, represented by the `flag` (289 vertices) mesh, whose rest state is flat. Fig.4 illustrates a few snapshots of a `flag` waving simulation, obtained by translating the right side of the `flag` mesh and anchoring the left side.

The interactive deformation of a complex hand (1515 vertices) model is illustrated in Fig.5, and a `torus` twist (576 vertices) is shown in Fig.6.

A sequence of deformations obtained by the DEND algorithm using E_T by free interactive shape-editing steps is shown in Fig.7 on a free bending of `dino` (10098 vertices) mesh. When dragging the handle vertices, the deformed surface should retain the look of the original surface in a natural way. The smooth regions of the surface should remain smooth, but if the surface contains some geometric details, as in the dinosaur tail and body, the shape and orientation of these details should be preserved. As shown from these preliminary examples, the DEND algorithm allows for a natural detail-preserving deformation.

7 LIMITS AND CONCLUSIONS

The main requirement for physically based surface deformation is an elastic energy that measures how much an object has been deformed from its initial configuration. In this paper a new constrained variational surface-based deformation model is proposed, by exploiting the total curvature as a better aesthetic measure for deformation of elastic bodies.

There are a number of crucial requirements on a shape deformation operation which make it a challenging problem: (i) the operation should be efficient enough for interactive work, (ii) it should provide local influence and detail preservation, (iii) the editing operation should naturally change the shape and simultaneously respect the structural detail, (iv) elastic bodies should not self-intersect as they deform.

The proposed variational model satisfies the requirements (ii) and (iii). The current version of the DEND algorithm in `Meshviz` does not allow for great performance, which will require an optimized implementation. Therefore, deformation editing can be achieved in real-time only for medium size objects like those used for the shown examples. Moreover, self intersections can be avoided by surrounding the surface of the object with bounding boxes and using collision dynamics. This is not yet available in our simple deformation software.

Nevertheless, we demonstrated that our deformation model yields realistic effects, while still maintaining computational tractability. Many open directions will be investigated, starting from considering total curvature energies that are minimized by a non-flat rest surface, following the more general energy in (1).

8 REFERENCES

- [1] Au O. K.-C., Fu H., Tai C.-L. and Cohen-Or D., *Handle-Aware Isolines for Scalable Shape Editing*, ACM Trans. on Graphics, 26/3, (2007).
- [2] Bendels G. H. and Klein R., *Mesh Forging: Editing of 3D-Meshes Using Implicitly Defined Occluders*, In Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, 207–217 (2003).
- [3] Botsch M. and Kobbelt L., *Real-time shape editing using radial basis functions*. Computer Graphics Forum (Proc. Eurographics), Vol.24/3, 611–621 (2005).
- [4] Botsch M., Kobbelt L., Pauly M., Alliez P., and Levy B., *Polygon Mesh Processing*. AK Peters - 2010.
- [5] Botsch M., Pauly M., Gross M. and Kobbelt L., *PriMo: Coupled Prisms for Intuitive Surface Modeling*. In Proc. of Eurographics symposium on Geometry Processing, 11–20 (2006).
- [6] Botsch M., Pauly M., Wicke M. and Gross M., *Adaptive Space Deformations Based on Rigid Cells*, Computer Graphics Forum, Vol. 26/3, 339–347 (2007).
- [7] Botsch M. and Olga Sorkine O., *On Linear Variational Surface Deformation Methods*, IEEE Trans. on Visualization and Computer Graphics, 14/1, 213–230 (2008).
- [8] Botsch M., Sumner R., Pauly M. and Gross M., *Deformation Transfer for Detail-Preserving Surface Editing*. In Proc. Vision, Modeling and Visualization, 357–364 (2006).
- [9] Clarenz U., U. Diewald, G. Dziuk, M. Rumpf, R. Rusu, *A finite element method for surface restoration with smooth boundary conditions*, Computer Aided Geometric Design **21**(5), 427–445 (2004).
- [10] Desbrun M., M. Meyer, P. Schroeder and A. Barr, *Implicit fairing of Irregular meshes using diffusion and curvature flow*, Computer Graphics (SIGGRAPH '99 Proceedings), 317–324 (1999).
- [11] do Carmo M.P., *Riemannian Geometry*, Birkhauser, Boston-Basel-Berlin, 1993.
- [12] Gain J. & Bechmann D., *A survey of spatial deformation from a user-centered perspective*. ACM Trans. on Graphics, 27/4, 107:1–107:21 (2008).
- [13] Griessmair J. & Purgathofer W., *Deformation of Solids with Trivariate B-Splines*. In Proceedings of Eurographics - EUROGRAPHICS '89, W. Hansmann, F.R.A. Hopgood and W. Strasser (Editors), Elsevier Science Publishers B.V. (North Holland), 137–148 (1989).
- [14] Helfrich W., *Elastic properties of lipid bilayers-theory and possible experiments*, Z. Naturforsch. C 28, 693-703 (1973).
- [15] Hsu W. M., Hughes J. F. & Kaufman H., *Direct manipulation of free-form deformations*. Proc. of ACM SIGGRAPH, New York, 177–184 (1992).
- [16] Huang J., Shi X., Liu X., Zhou K., Wei L.-Y., Teng S., Bao H., Guo B. & Shum H.-Y., *Subspace Gradient Domain Mesh Deformation*. ACM Trans. on Graphics, Vol. 25/3, 1126–1134 (2006).
- [17] Lamousin H. J. & N. N. Waggenspack N. N., *NURBS-based free-form deformations*. IEEE Computer Graphics and Applications, Vol. 14/6, 59–65 (1994).
- [18] Lipman Y., Sorkine O., Cohen-Or D., Levin D., Rössl C., Seidel H.-P., *Differential coordinates for interactive mesh editing*, In Proceedings of Shape Modeling International, IEEE Computer Society Press, 181–190 (2004).
- [19] Meyer M., M. Desbrun, P. Schroeder, P., and A. Barr, *Discrete Differential Geometry Operators for Triangulated 2-Manifolds*, in: Proc. VisMath '02, Berlin-Dahlem, Germany, 237–242, (2002).
- [20] Morigi S., *Geometric Surface Evolution with Tangential Contribution*, Journal of Computational and Applied Mathematics; 233, 1277–1287 (2010).
- [21] Osher S., R. Fedkiw, *Level set methods and dynamic implicit surfaces*. Eds. S.S.Antman, J.E. Marsden, L. Sirovich - Springer Verlag, New

York, 2003.

- [22] Pauly M., Keiser R., Kobbelt L. & Gross M., *Shape Modeling with Point-Sampled Geometry*. ACM Trans. on Graphics, Vol. 22/3, 641–650 (2003).
- [23] Sederberg T. W. & Parry S. R., *Free-form deformation of solid geometric models*. Computer Graphics, 20/4, 151–160 (1986).
- [24] Sheffer A., & Kraevoy V., *Pyramid coordinates for morphing and deformation*. In Second International Symposium on 3D Data Processing, Visualization and Transmission, 68–75 (2004).
- [25] Shi X., Zhou K., Tong Y., Desbrun M., Bao H. & Guo B., *Mesh Puppetry: Cascading Optimization of Mesh Deformation with Inverse Kinematics*. ACM Trans. on Graphics, Vol. 26/3 (2007).
- [26] Sorkine O., Cohen-Or D., Lipman Y., Alexa M., Rössl C. & Seidel H.-P., *Laplacian Surface Editing*. Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, ACM Press, 179–188 (2004).
- [27] Sieger D., Menzel S., & Botsch M., *A Comprehensive Comparison of Shape Deformation Methods in Evolutionary Design Optimization*. Proceedings of International Conference on Engineering Optimization - 2012.
- [28] Sumner R. w., Schmid J. & Pauly M., *Embedded Deformation for Shape Manipulation*. ACM Trans. on Graphics, Vol 26/3, 80:1–80:7 (2007).
- [29] Surazhsky T. and E.Magid and O.Soldea and G.Elber and E. Rivlin, *A comparison of Gaussian and mean curvatures estimation methods on triangular meshes*, In Proceeding ICRA 2003, 1021–1026 (2003).
- [30] Terzopoulos D., K. Fleischer, *Deformable models*, The Visual Computer, Vol 4, 306–331 (1988).
- [31] Terzopoulos D., *On matching deformable models to images*, Optical Society of America, Topical Meeting on Machine Vision, 160–163 (1986).
- [32] Terzopoulos D., Platt J., Barr A. & Fleischer K., *Elastically Deformable Models*. ACM Siggraph Computer Graphics, Vol.21/4, 205–214 (1987).
- [33] Willmore T.J., *Total Curvature in Riemannian Geometry* (John Wiley and Sons, New York, 1982).
- [34] Yoshizawa S., *Fair Triangle Mesh Generation with Discrete Elastica*, Geometric Modeling and Processing, 119–123 (2002).
- [35] Yu Y., Zhou K., Xu D., Shi X., Bao H., Guo B. & Shum H.-Y., *Mesh Editing with Poisson-Based Gradient Field Manipulation*. ACM Trans. on Graphics, 23/3, 641–648 (2004).

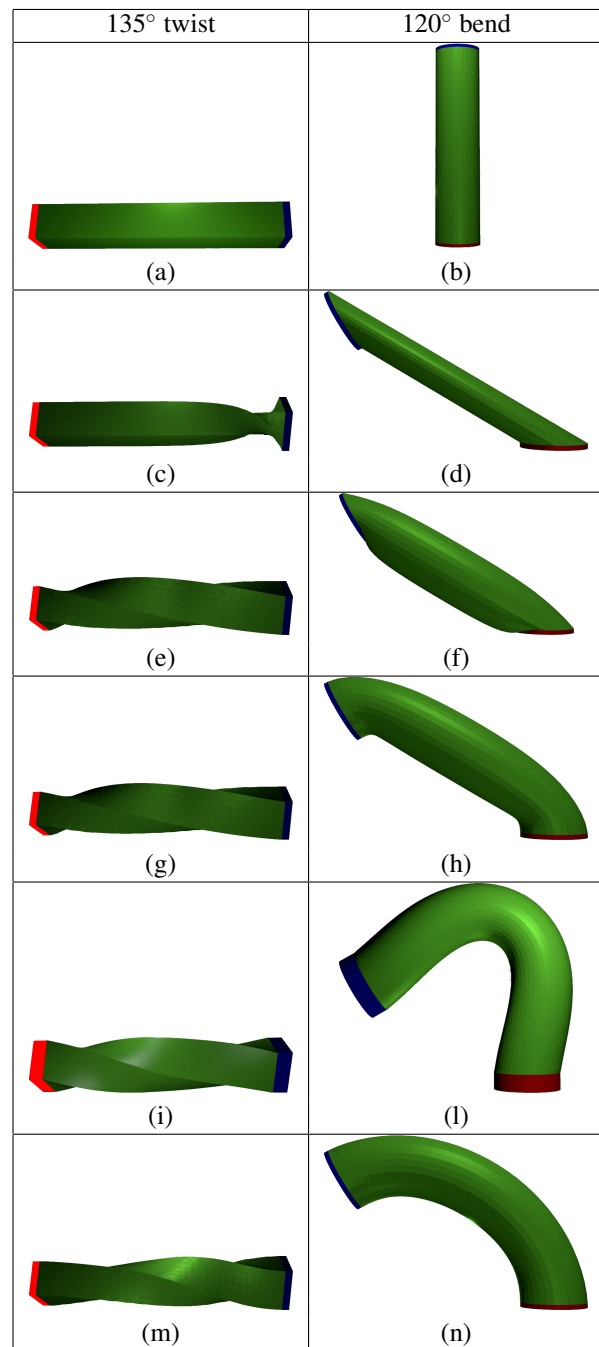


Figure 1: Deformation of the bar (left column) and cylinder (right column) meshes. The deformations are achieved by anchoring the red vertices and twisting/bending the blue vertices.

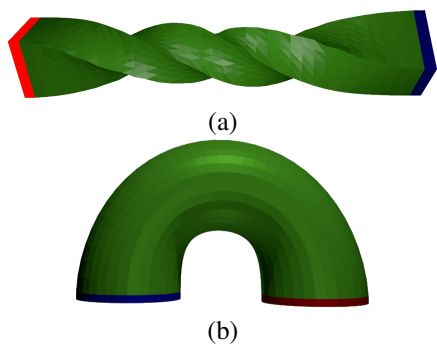


Figure 2: (a) 300° twist of the bar mesh; (b) 180° bend of the cylinder mesh.

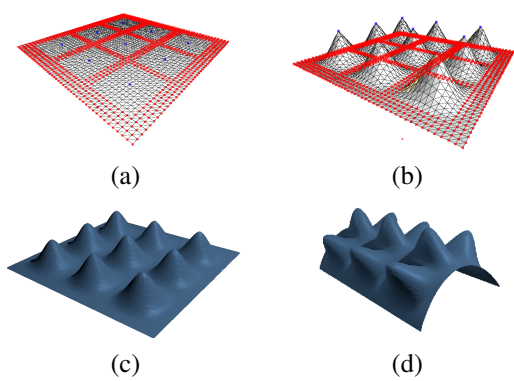


Figure 3: (a) Original plane model; (b) Bumpy plane achieved by deforming (a) anchoring the red vertices and translating the blue vertices; (c) undeformed shaded model; (d) deformation by bending the two sides of the mesh.

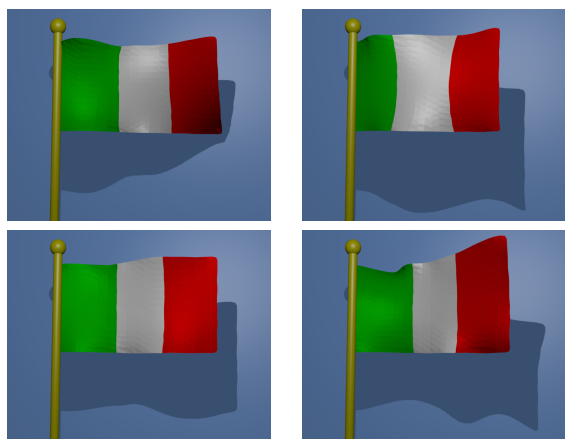


Figure 4: A sequence of flag deformations by translating the right side and anchoring the left side of the mesh.

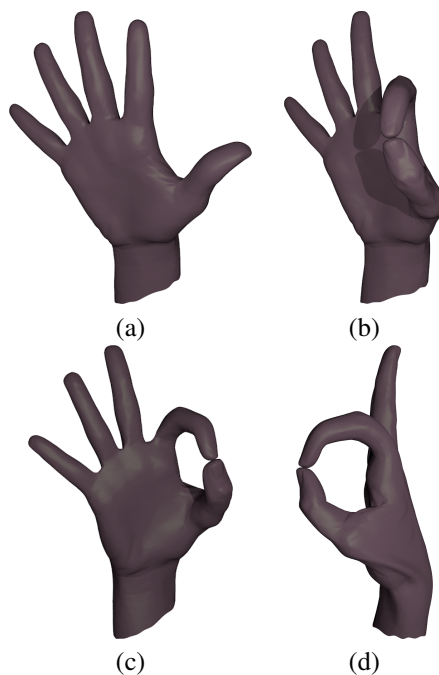


Figure 5: Interactive deformation of the hand mesh: (a) undeformed mesh; (b)-(c)-(d) deformed mesh from different points of view.

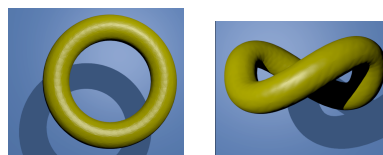


Figure 6: A twist of torus mesh.

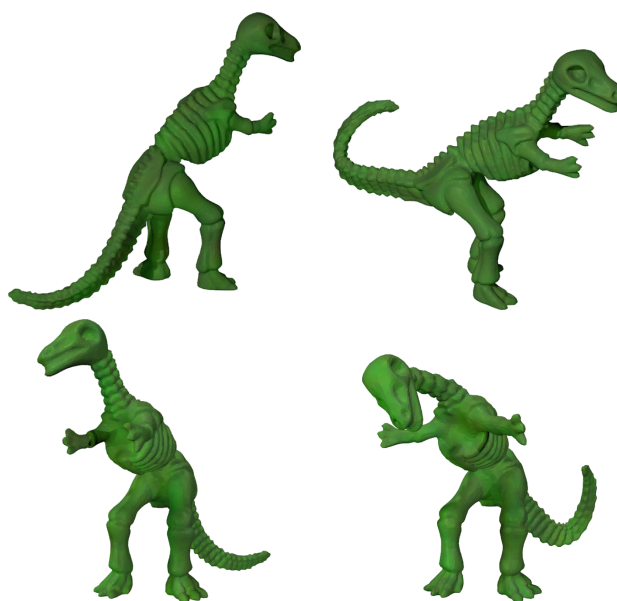


Figure 7: A sequence of dino deformations by free interactive shape-editing steps.

Interactive Radial Volume Histogram Stacks for Visualization of Kidneys from CT and MRI

M. Alper Selver

Dokuz Eylül University
Dept. of Electrical and Electronics
Engineering, Tinaztepe Campus
35160, Buca, Izmir, TURKEY

alper.selver@deu.edu.tr

Merve Özdemir

Dokuz Eylül University
Institute of Natural and Applied
Sciences, Tinaztepe Campus
35160, Buca, Izmir, TURKEY

merveozdemir89@gmail.com

Eşref Selvi

Dokuz Eylül University
Institute of Natural and Applied
Sciences, Tinaztepe Campus
35160, Buca, Izmir, TURKEY

esref.selvi@hotmail.com

ABSTRACT

Optical parameter assignment via Transfer Functions (TF) is the sole interactive part in medical visualization via volume rendering. Being an interactive element of the rendering pipeline, TF specification has very important effects on the quality of volume-rendered medical images. However, TF specification should be supported by informative search spaces, interactive data exploration tools and intuitive user interfaces. Due to the trade-off between user control and TF domain complexity, integrating different features into the TF without losing user interaction is a challenging task since both are needed to fulfill the expectations of a physician. By addressing this problem, we introduce a semi-automatic method for initial generation of TFs. The proposed method extends the concept of recently introduced Volume Histogram Stack (VHS), which is a new domain constructed by aligning the histograms of the image slices of a CT and/or MR series. In this study, the VHS concept is extended by allowing the user to define an alignment axis using orthogonal multi planar reconstructions via simple, yet effective, interaction mechanisms. The construction of VHS according to the slices generated specifically for user defined search space allows the more informative integration of local intensity distribution, and better spatial positioning of the organ of interest into the TF. For testing, the proposed strategy is applied to kidney visualization from CT and MRI series. The performance of extended VHS domain is evaluated via intensity based TF design. Volumetric histogram based manual TF specifications are quantitatively compared to VHS based manual tweaking of original slices, and to extended-VHS based automatic TF design. The results show both quantitatively and qualitatively enhanced rendering quality for kidney visualization.

Keywords

Transfer Function Specification, Volume Rendering, Medical Visualization, Kidney.

INTRODUCTION

Visualization aims to produce clear and informative pictures of the important structures in a data set. Depending on the application, this requires interactive determination of visual parameters such as opacity and color. In volume rendering technique [Dre98], combinations of these visual parameters can be determined during the rendering pipeline. During the generation of volume rendered images, Transfer Function (TF) specification is the step where these adjustments can be done. Therefore, it is crucial and important to design accurate TFs to produce meaningful and intelligible 3-D images. However, TF design is a very difficult task because of the availability of various possibilities in extensive search spaces of TFs [Pfi00]. Since this flexibility of search space cannot be kept in strict bounds,

specification of an appropriate TF is a challenging problem, where effective initial TF designs should be generated prior to the optimization part, which is controlled by the user. Moreover, advanced user interaction interfaces [Rez06] and data exploration tools [Sel07] should be provided for fulfilling user expectations.

To overcome the difficulty of initial TF generation generally a number of predefined TF presets are used as starting point (so called initial TF design). The main idea behind this approach is that certain types of volume data are standardized in the range of data values and special sub-ranges are assigned to the same type of structure (thus, predefined TFs are adjusted due to these ranges). However, volumetric data usually have varying characteristics even in different samples of the same application. For instance, in medical imaging, depending on different modality settings, injection of a contrast media or environmental circumstances, the sub-ranges of the tissues may vary significantly. For these reasons, a limited number of TF presets cannot be enough to cover all possible cases and to provide useful initial

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

TFs. In order to create a useful initial TF that provides a good basis prior to optimization, an automatic sub-range detection method that finds the intensity range for each structure of interest is needed. Moreover, it is necessary to integrate the developed method into the TF design procedure without losing user control and interaction over the search space.

TF specification methods can roughly be divided into two groups as data-centric [Kin98] and image-centric [Shi98]. In both of them, finding the contours that are hidden behind another is an important aspect of useful TF generation [Baj97]. To achieve this goal, effective use of spatial information is necessary [Roe05]. In [Roe05], spatial TFs are introduced as 1D or multidimensional TFs, where spatial information has been used to derive the color, whereas statistical (and/or spatial information) is used to set up the opacity. Local properties are used to increase the performance of topological approaches. In [Sat00], 3D filters, based on gradient vector and Hessian matrix, are used to enhance specific 3D local intensity structures. In [Lun06a], histogram contents for local neighborhoods are used to detect and separate tissues with similar intensities. In [Lun06b], an enhancement that amplifies ranges corresponding to spatially coherent materials by using alpha-histograms, which are individually retrieved by dividing the data set into local regions, is implemented. These studies show the importance of local information in solving major problems in TF generation such as the classification of overlapping tissues. Recent studies also focus on size [Wes10], shape, appearance [Saa10] and visibility [Car11] of anatomical structures for constructing effective TFs.

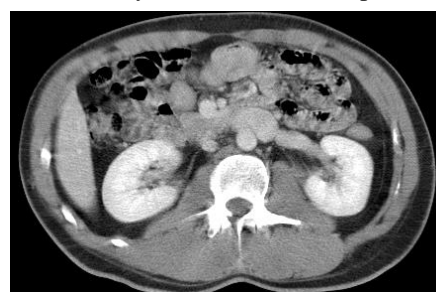
By addressing this problem, we introduce a semi-automatic method for initial generation of TFs. The proposed method extends the concept of Volume Histogram Stack (VHS). VHS is recently introduced in [Sel09] as a new domain which is created by aligning the histograms of the image slices of a CT/MR series. Histograms were generated from orthogonal directions of slice planes, namely, axial, coronal and sagittal. Thus, VHS can represent the intensity values of the tissues as well as their spatial information and local distributions (via lobes in VHS) which are not available in conventional volume histograms. The tissues which are at different slices but with similar gray level distributions can clearly be distinguished by using this spatial information. Then, a tissue (a structure of interest) can effectively be visualized by determining its corresponding lobe(s) in VHS, which represents that structure of interest, and by assigning a color-opacity value to that lobe.

In this study, the VHS concept is extended by allowing user to define an alignment axis using

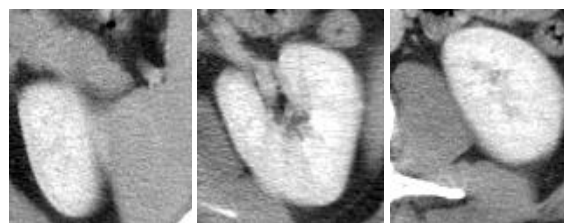
orthogonal multi planar reconstructions via simple, yet effective, interaction mechanisms. The construction of VHS according to aligning the slices generated specifically for user defined search space allows the integration of local intensity distribution, and spatial positioning of the organ of interest into the TF. For testing, the proposed strategy is applied to kidney visualization from CT and MRI series. The performance of extended VHS domain is evaluated via intensity based TF design. Volumetric histogram based manual TF specifications are quantitatively compared to VHS based manual tweaking of original slices, and extended VHS based automatic TF design. The results show both quantitatively and qualitatively enhanced rendering quality for kidney visualization. With the help of this expansion, the VHS becomes an effective new domain as a search space for TF specification on any kind of 3D data.

DATA SETS

The first application is an abdominal CT series taken at the venous phase for the evaluation of a liver transplantation donor [Sel07] (Figure 1). Images were acquired after contrast agent injection at portal phase using a Philips Secura CT with two detectors equipped with the spiral CTA option and located in Dokuz Eylül University Radiology Department. The second data set is acquired using a 1.5T MRI system located in the same department. Both scanners produce 12 bit DICOM images with a resolution of 512×512 for CT and 256×256 for MR. The data sets were collected from the Picture Archiving and Communication System of the same department.



(a)



(b)

Figure 1. a) An original CT image slice (direction of acquisition), (b) Left kidneys at different slices (left to right order is from beginning to the end of the series). Note that, kidney start to occur smaller, becomes larger and gets smaller again.

METHODOLOGY

In CT and/or MR data sets, the kidneys may have different gray value distributions according to environmental circumstances, injection of a contrast media, and certain modality parameters. Moreover, their location, orientation and size may differ due to patient anatomy (Figure 1). Although, there is a calibrated intensity scale in CT (i.e. Hounsfield Units – HU), the above mentioned diversity still exists. Moreover, volume rendering is not commonly used for MR data sets since there is no calibrated intensity scale. In conventional approach, both for CT and MR data sets, the volume histogram is the main guide to find the tissues of interest.

On the other hand, these kidneys do not always correspond to visible peaks as reported in [Lun06a]. The reason behind this is the existence of dominant peaks which occur due to unified intensity range of overlapping tissues such as liver and spleen. Especially in CT, these abdominal organs occur in a very narrow range of HU values. This overlap hardens the usage of TFs in the visualization of kidneys.

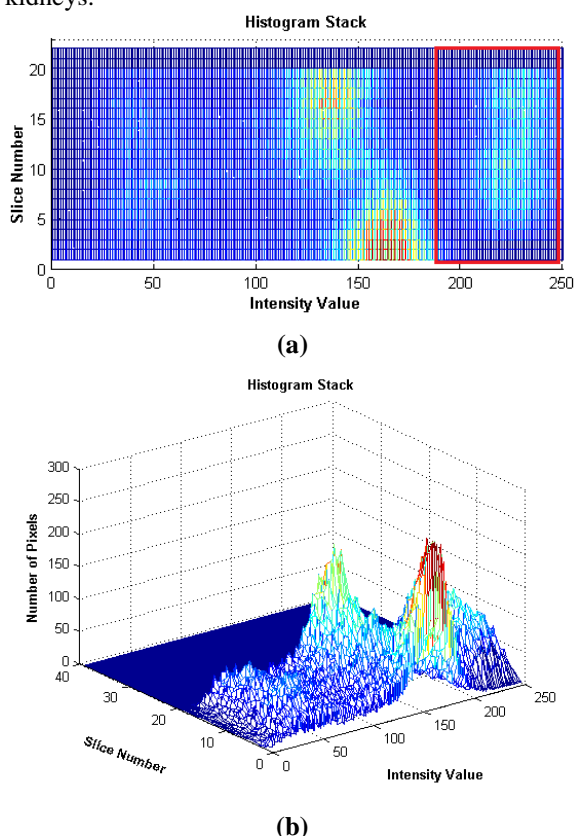


Figure 2. Left kidney analysis from CT (a) VHS for cropped axial slices (top view), red rectangle shows the intensity range of left kidney (b) VHS for axial slices (rear view).

Through a series of abdominal image slices, kidneys begin to occur as small objects at first, expand in the successive slices, and finally disappear by becoming

smaller objects as slices proceed (Figure 1.b). This causes a lobe-like histogram distribution for kidneys. The peaks of the lobes are at the slices in which kidneys appear biggest in size (Figure 2). This additional information is available only if the z-dimension (orthogonal to the slices) is used as exploited in the VHS concept.

This provides the advantage of discriminating kidneys from other organs that are spatially separated in the direction of acquisition even if the intensity range of these organs/tissues completely overlap with kidneys. Traditional volume histograms cannot take this advantage as they represent cumulative gray-level distribution over all data set. On the other hand, in practice, kidneys do not get completely separate lobes. Instead, the lobes have intersecting regions if kidneys and similar intensity organs are partially spatially non-separated.

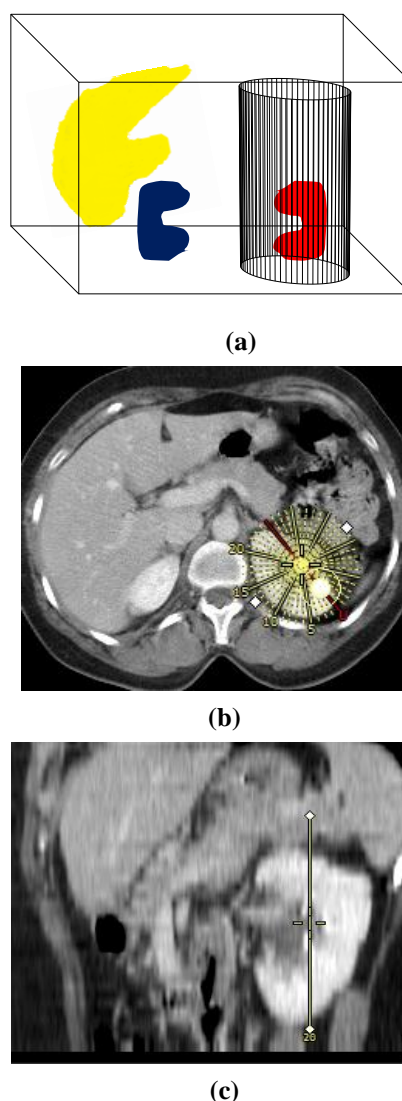


Figure 3. (a) Reconstruction strategy inside the volume, (b) circular area selection from axial, (c) vertical height selection from sagittal (left kidney)

To take advantage of spatial knowledge more, VHS approach has been extended to produce alignments through x and y-dimensions. For instance, in the above mentioned case, if the major slicing axis is z-dimension, VHS which can be generated using the histograms calculated for axial images (x-y dimensions) and aligned through z-dimension to construct VHS. Moreover, with the help of spatial extension, VHS can also be generated by using the histograms of coronal images (y-z dimensions) and aligning them through x-dimension or the histograms of sagittal images (x-z dimensions) aligning them through y-dimension. This enables organ based selection for the VHS which can be generated based on the organ to be visualized and independent of slicing axis without an additional scanning procedure. With this opportunity, VHS can distinguish structures which are separated in any of x, y, and z-dimensions.

In this study, this approach is further extended for kidney visualization by allowing the user to select a cylindrical tube using a Multi Planar Reconstruction (MPR) interface (Figure 3.a). The interaction mechanism enables user to select a circular Region of Interest (ROI) first (Figure 3.b), in any one of three MPR reconstructions (i.e. axial, sagittal, and coronal). This step determines the center and radius of the 3-D cylinder. Then, using an MPR image, which is orthogonal to the MPR image used in the first step, the height of the cylinder is determined (Figure 3.c). Finally, based on a defined thickness, new image slices, all of which have same center position and size of $(2 \times \text{radius}) \times \text{height}$, are generated (Figure 4.a). The VHS generated using these user defined slices by aligning their histograms exploits more a priori information as providing kidney specific inter-slice spatial domain knowledge (Figure 4.b-c). Thus, the information on local histogram distributions of organ of interest is more evident (Figure 5-6). The tissues which are at different slices but with similar gray-level distributions can clearly be distinguished by using this spatial information.

The VHS data exploits more a priori information as saving inter-slice spatial domain knowledge since each slice histogram is represented separately. It demonstrates changes in the gray values through the series of slices, thus includes information on local histogram distributions of tissues. For example, when a tissue appears larger in an image, the number of pixels representing this tissue also increases and vice versa. The VHS demonstrates these changes much better than the volume histogram since the data distribution is shown in a continuous way through the series. Thus, it can represent the intensity values of the tissues as well as their spatial information and local distributions which are not available in conventional volume histograms.

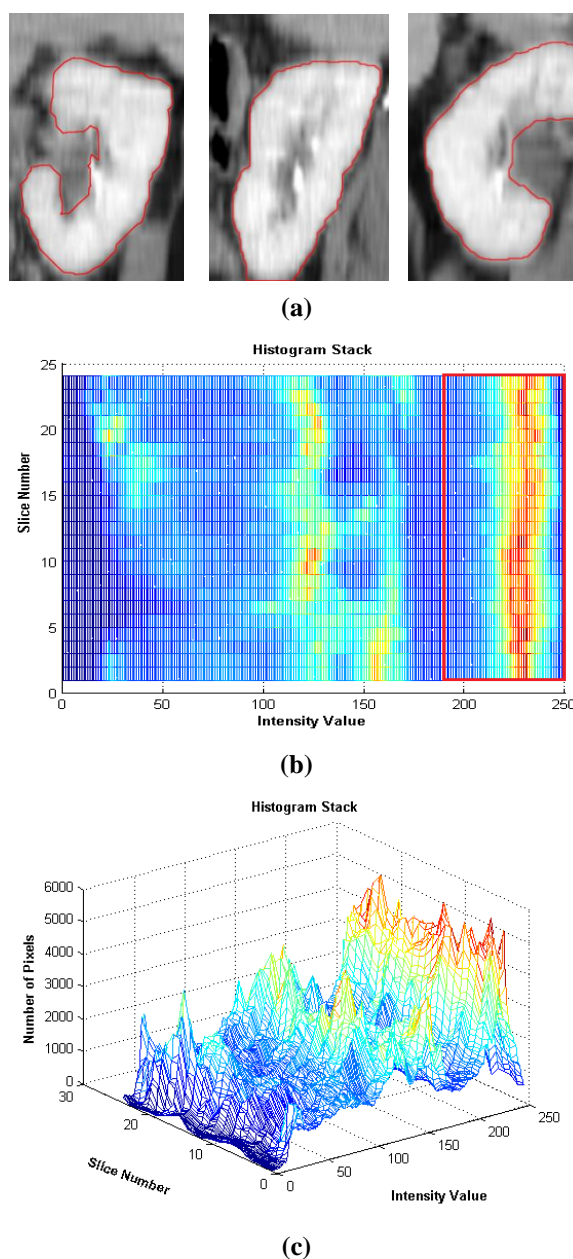


Figure 4. (a) Left kidneys at reconstructed slices (b) VHS for reconstructed slices (top view), red rectangle shows the intensity range of left kidney (c) VHS for reconstructed slices (rear view).

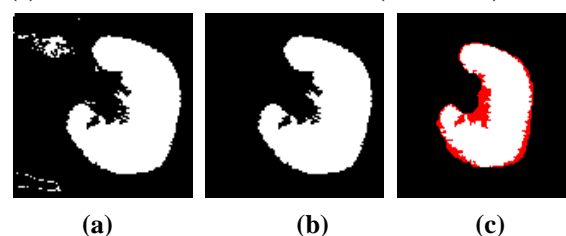


Figure 5. Axial CT Left Kidney (LK), (a) result of thresholding $185 < T < 250$, (b) result of post-processing, (c) difference with reference.

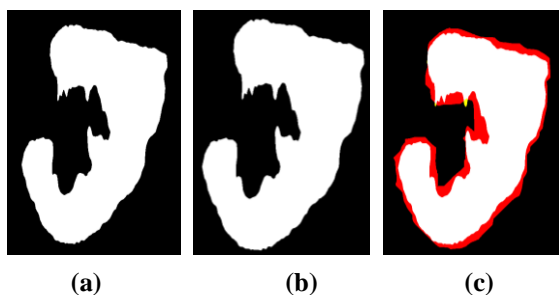


Figure 6. Reconstructed CT Left Kidney (LK), (a) result of thresholding $200 < T < 250$, (b) result of post-processing, (c) difference with reference.

RESULTS

We have tested our algorithm on both MR and CT data sets. Four data sets (i.e. 2 MR and 2 CT) consist of axial DICOM slices from abdomen, thus include both left and right kidneys. First, we obtained the VHS of both axial and reconstructed images of left and right kidneys. Then, for both VHS, a threshold range is determined manually. According to the indicated threshold range, an intensity based TF is designed and pixels are classified. A simple post-processing part has been used which uses “connected component” analysis. With the help of post-processing, small misclassifications (Figure 10. b, f,

j, n, s, w) are eliminated by selecting largest connected component (Figure 10. c, g, k, o, t, y).

Accuracy of classifications is compared with reference data sets, which were manually segmented. The same procedure is also applied to axial data sets and results are compared. Considering each pixel is either assigned to kidney (i.e. Positive - 1) or not (Negative - 0), the comparisons are made by counting True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) instances. The error measures calculated between classification results and reference data using above variables are:

- 1) False Positive Ratio ($FPR = 100 * FP / (TN + FP)$),
- 2) False Negative Ratio ($FNR = 100 * FN / (TP + FN)$),
- 3) Sensitivity, which gives the percentage of positive labeled instances that were predicted as positive, and calculated as ($SE = 100 * TP / (TP + FN)$),
- 4) Specificity, which gives the percentage of negative labeled instances that were predicted as negative and calculated as ($SP = 100 * TN / (TN + FP)$),
- 5) Positive Predictive Value (i.e. Precision), which gives the percentage of positive predictions that are correct and calculated as ($PPV = 100 * TP / (TP + FP)$),
- 6) Negative Predictive Value, which gives the percentage of negative predictions that are correct and calculated as ($NPV = 100 * TN / (TN + FN)$).

The values of these measures are given in Tables 1-3.

	Organ	Data Type	FPR	FNR	SE	SP	PPV	NPV
CT Data set	LK	Axial	0.17	14.65	85.35	99.72	99.58	89.82
		Reconst.	0.23	15.86	84.14	99.70	99.13	93.80
	RK	Axial	0.52	13.27	86.73	99.27	97.96	95.05
		Reconst.	0.49	12.56	87.44	99.28	98.50	93.76
MR Data set	LK	Axial	10.29	28.03	71.97	91.21	85.15	88.29
		Reconst.	2.51	18.98	81.02	96.36	93.69	89.44
	RK	Axial	0.08	20.46	79.54	99.89	99.71	91.52
		Reconst.	0.53	9.50	90.50	99.06	98.81	92.40

Table 1. Kidneys Results for Axial and Reconstructed CT and MR Data set without post-processing

	Organ	Data Type	FPR	FNR	SE	SP	PPV	NPV
CT Data set	LK	Axial	6.63	25.97	74.03	90.47	85.70	81.80
		Reconst.	1.57	15.76	84.24	97.92	91.73	93.75
	RK	Axial	4.60	13.08	86.92	94.56	85.54	94.89
		Reconst.	5.13	11.64	88.36	93.23	88.01	93.74
MR Data set	LK	Axial	13.15	27.81	72.19	87.32	79.83	87.97
		Reconst.	14.63	18.96	81.04	81.31	73.89	88.01
	RK	Axial	1.55	19.63	80.37	97.85	94.79	91.63
		Reconst.	0.74	9.38	90.62	98.71	98.32	92.46

Table 2. Kidneys Results for Axial and Reconstructed CT and MR Data set with post-processing

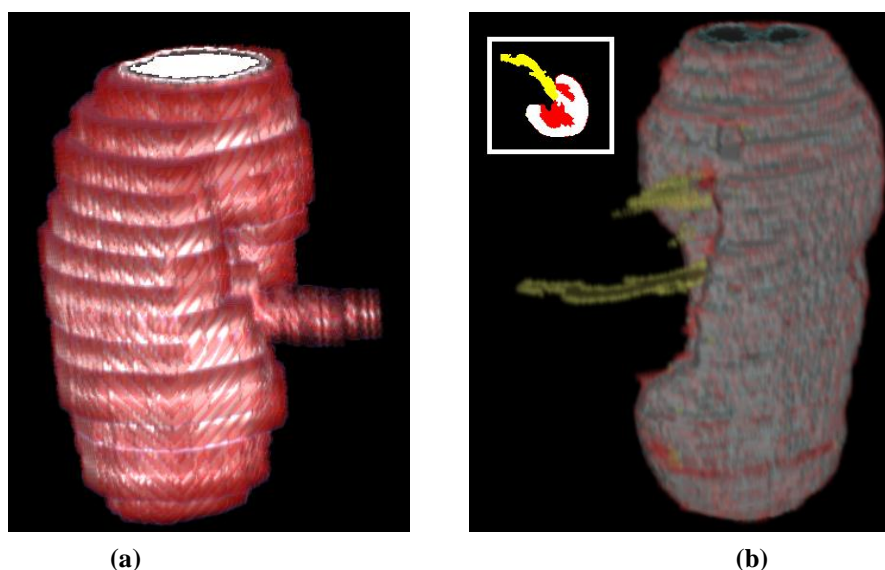


Figure 7. 3D reconstruction of axial image of (a) MR right kidney, (b) CT left kidney.

In Table 1-2, the results of first CT - MR data set pair are given for axial and reconstructed images. These data sets differ from the second CT - MR data set pair in terms of image contrast. In other words, histogram of first CT-MR data set pair is narrower than the histogram of second CT-MR data set pair.

In Table 1, application of proposed algorithm without post-processing is given, while Table 2 covers results with post-processing.

The results show comparable performance for left kidney (LK) of axial and reconstructed VHS for CT data. On the other hand, results for MR data show improved performance especially when no post-processing is done. This is an important result since it shows better data classification on reconstructed TF domain than axial TF domain.

Considering right kidney (RK) (i.e. 3rd and 4th rows), the results show that FPR values are comparable but FNR measures are significantly better using reconstructed VHS for CT data. The results for MR data set (i.e. 7th and 8th rows) are. Similar to the results in CT, error measures show significantly improved FP and FN rates using reconstructed VHS domain.

The selection of the orientation and size of radial reconstruction for VHS is shown in Figure 8. In Figure 9, VHS for both axial and reconstructed data and manually determined threshold ranges are shown. Figure 10 shows slice-by-slice results for a set of selected images.

Table 3 shows the results of error measures for second CT - MR data set pair. For CT data, FNR rate is significantly reduced for both LK and RK using reconstructed VHS domain. Moreover, for MR data set, both FPR and FNR are lower when

reconstructed VHS is used. Other measures also reflect the superior performance of proposed domain compared to the original domain. Figure 11 shows slice-by-slice results for a set of selected images for these data sets. Two exemplary 3D reconstructions are given in Figure 7 (note that no-post processing filters are used to increase rendering quality to show FP and FN voxels clearly).

CONCLUSIONS

This study presents a new domain for TF specification by extending volume histogram stacks, which are constructed by aligning histogram of each slice in a medical image series. The extension is done by using multi planar reconstructions and a user-friendly interface to determine a radial reconstruction of the original data. Although four data sets are used, the intensity TF based classification using reconstructed VHS is shown to have higher performance than using VHS for axial images. Although, these results should be confirmed by increasing number of applications, some early conclusions can be made. First, the shape of kidneys show less variability in size at reconstructed images compared to axial slices, which is an important advantage for shape based approaches. Second, in this study, reconstructed VHS are created using interpolated images (i.e. Figure 4.a, 10.e, m, v, 11.f, p, aa). However, using data without interpolation might increase the performance.

An automatic method finding appropriate intensity range of an organ of interest especially using local VHS information is a challenging future study. Also, a user interface, which can use different geometries for VHS generation, will be developed for allowing the use of VHS for other organs.

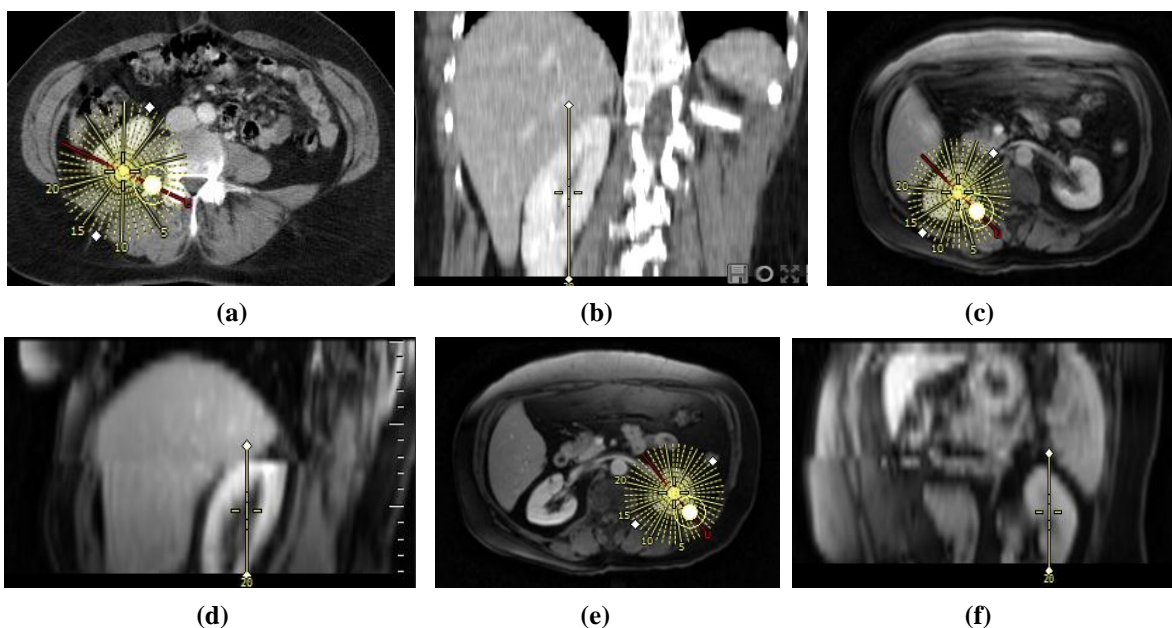


Figure 8. Circular area and vertical height selections (a, b) CT RK, (c, d) MR RK, (e, f) MR LK

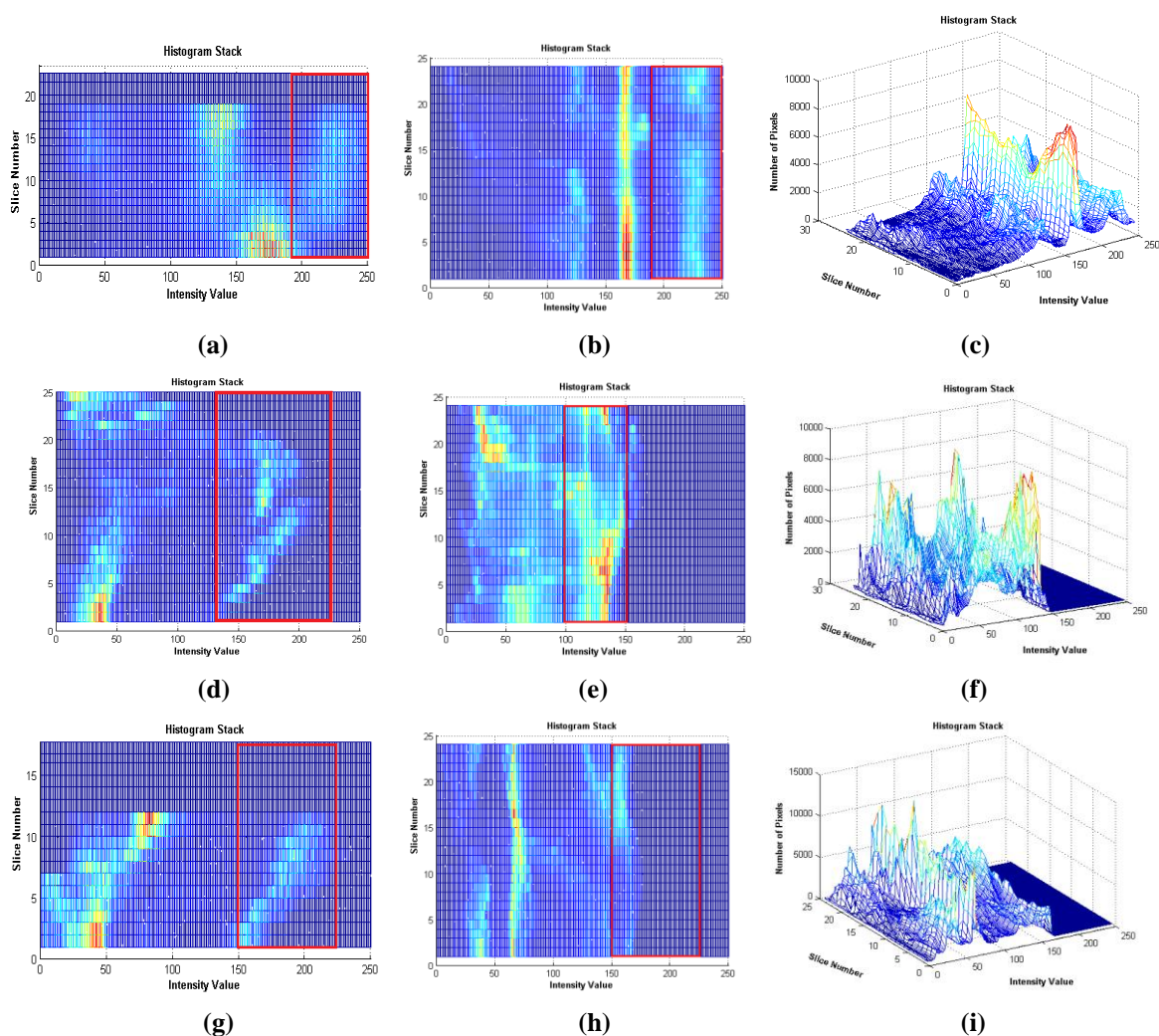


Figure 9. Right kidney analysis from CT (a) VHS for axial slices (top view), VHS for reconstructed slices, (b) top view, (c) side view. Left kidney analysis from MR (d) VHS for axial slices (top view), VHS for reconstructed slices, (e) top view, (f) side view. Right kidney analysis from MR (g) VHS for axial slices (top view), VHS for reconstructed slices, (h) top view, (i) side view. (PS: red rectangles show the intensity ranges of kidneys).

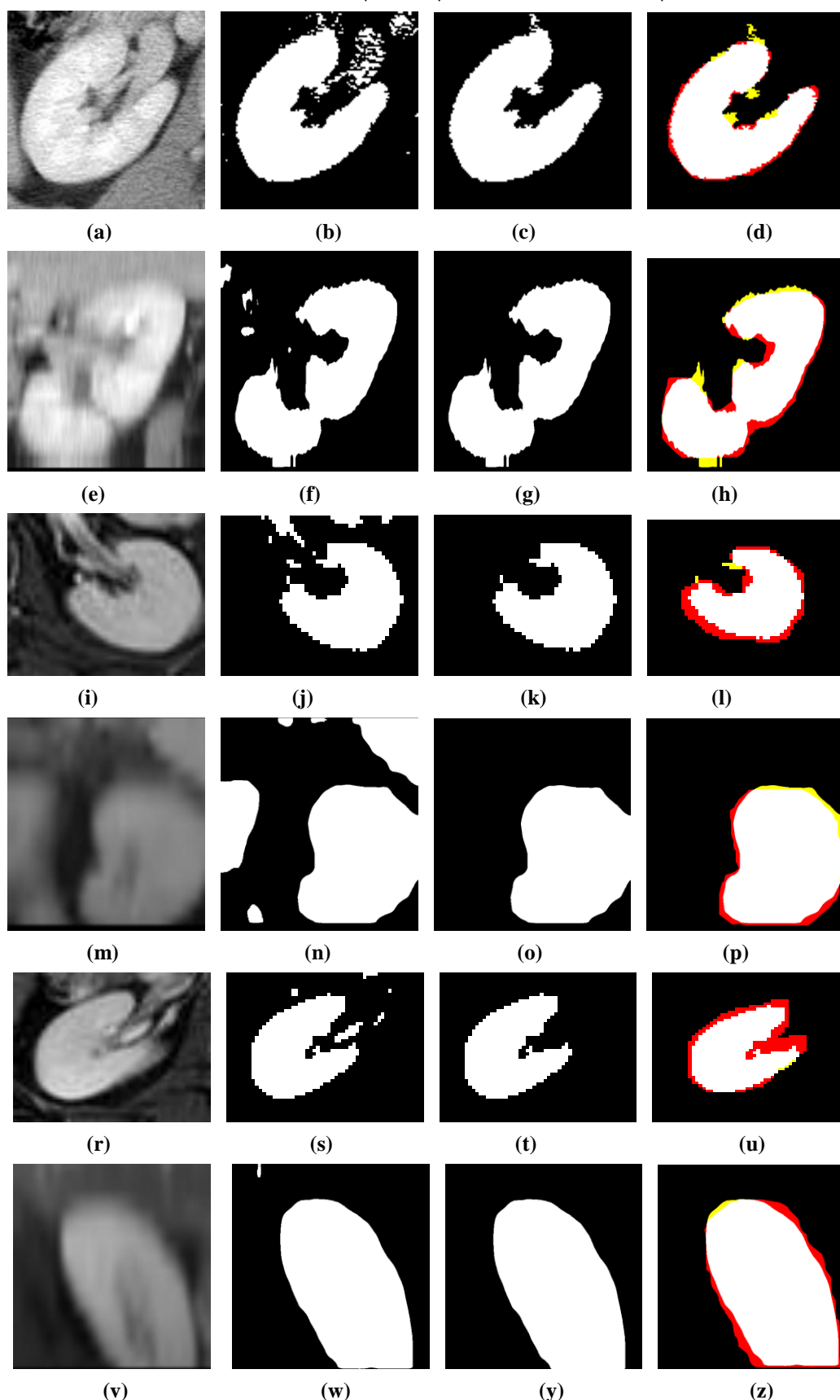


Figure 10. CT Right Kidney (RK) (a) cropped axial image, (b) result of thresholding $185 < T < 250$, (c) result of post-processing, (d) difference with reference, (e) reconstructed image, (f) result of thresholding $185 < T < 250$, (g) result of post-processing, (h) difference with reference. MR Left Kidney (LK) (i) cropped axial image, (j) result of thresholding $140 < T < 230$, (k) result of post-processing, (l) difference with reference, (m) reconstructed image, (n) result of thresholding $100 < T < 150$, (o) result of post-processing, (p) difference with reference, MR Right Kidney (RK) (r) cropped axial image, (s) result of thresholding $150 < T < 230$, (t) result of post-processing, (u) difference with reference, (v) reconstructed image, (w) result of thresholding $100 < T < 180$, (y) result of post-processing, (z) difference with reference.

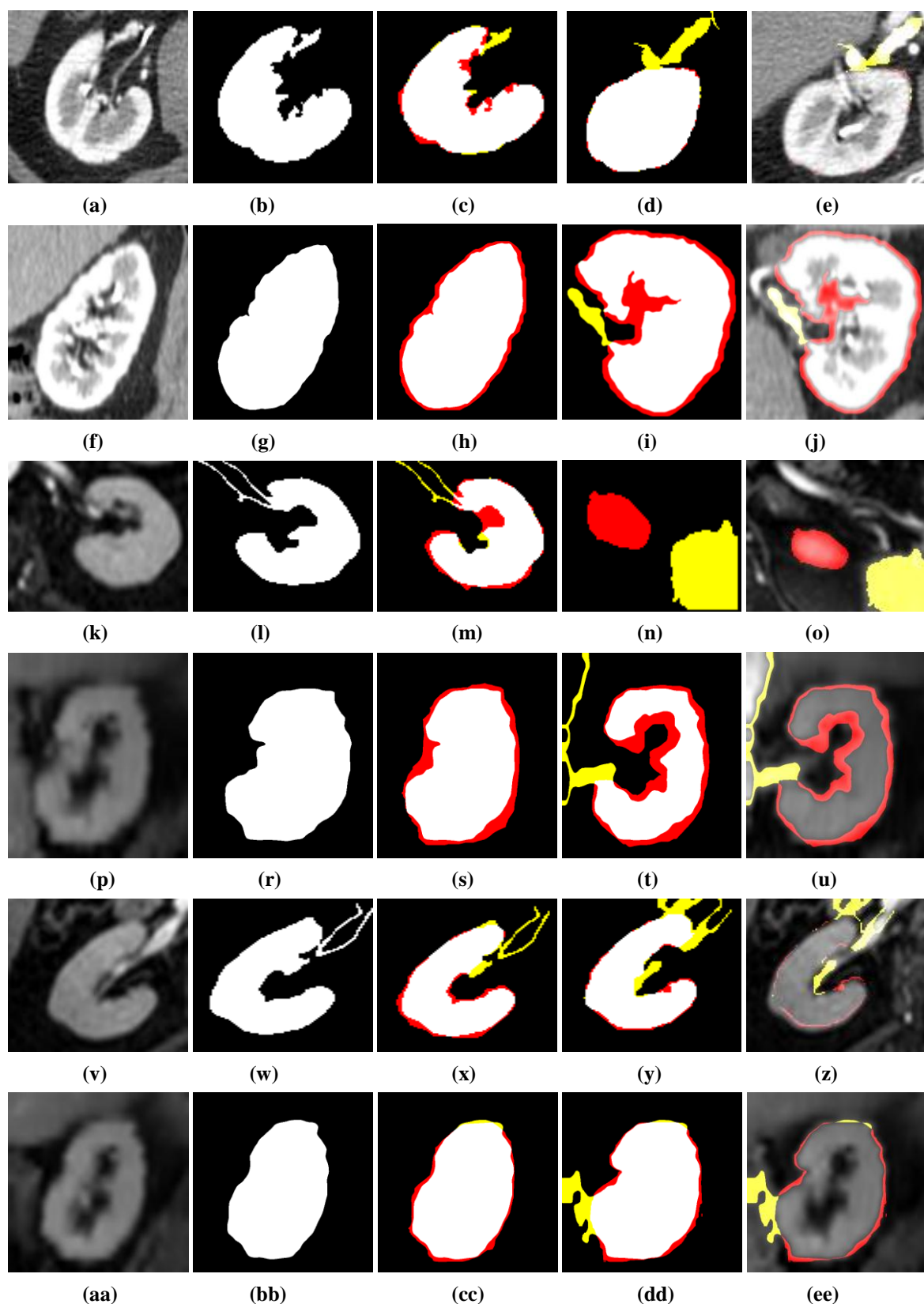


Figure 11. For all rows of the figure, from left to right, the images show: cropped axial or reconstructed image, result of post-processing, difference of result with reference, another example of difference image from the same data set, transparent illustration of FP and FN pixels on the image. The rows of the figure correspond to (a)-(e) axial CT Right Kidney, (f)-(j) reconstructed CT Right Kidney, (k)-(o) axial MR Left Kidney, (p)-(u) reconstructed MR Left Kidney, (v)-(z) axial MR Right Kidney, (aa)-(ee) reconstructed MR Right Kidney (FP pixels are shown in yellow while FN pixels are shown in red).

	Organ	Data Type	FPR	FNR	SE	SP	PPV	NPV
CT Data set	LK	Axial	0.39	22.43	77.57	99.55	98.17	94.47
		Reconst.	0.75	4.4	95.6	99.1	96.72	98.73
	RK	Axial	0.86	8.13	91.87	98.9	96.7	97.83
		Reconst.	2.89	5.62	94.38	97.06	88.54	97.31
MR Data set	LK	Axial	1.05	22.32	77.68	98.99	93.06	96.31
		Reconst.	0.18	22.69	77.31	99.77	99.43	89.77
	RK	Axial	0.78	10.78	89.22	99.02	96.83	97.51
		Reconst.	0.58	10.48	89.52	99.17	98.38	94.81

Table 3. Kidneys Results for Axial and Reconstructed CT and MR Data set with post-processing

ACKNOWLEDGMENTS

This study is supported by TUBITAK EEEAG under grant number 112E032.

REFERENCES

- [Baj97] Bajaj, C.L., Pascucci, V., Schikore, D.R., "The Contour Spectrum," Proc. Eighth IEEE Visualization Conf. (VIS '97), pp. 167-173, 1997.
- [Car11] Carlos D. C., Kwan-L. M., "Visibility Histograms and Visibility-Driven Transfer Functions," IEEE Transactions on Visualization and Computer Graphics, 17(2), 192-204, 2011.
- [Dre98] Drebin RA, Carpenter L, Hanrahan P, "Volume Rendering," Proc. ACM SIGGRAPH '88, pp. 65-74, 1988.
- [Kin98] Kindlmann, G., Durkin, J.W., "Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering," Proc. Ninth IEEE Visualization Conf. (VIS '98), pp. 79-86, 1998.
- [Lun06a] Lundström, C., Ljung, P., Ynnerman, A., "Local Histograms for Design of Transfer Functions in Direct Volume Rendering," IEEE Transaction on Visualization and Computer Graphics, vol. 12(6), 1570-1579, Nov./Dec. 2006.
- [Lun06b] Lundström, C., Ynnerman, A., Ljung, P., Persson, A., Knutsson, H., "The Alpha-Histogram: Using Spatial Coherence to Enhance Histograms and Transfer Function Design," Proc. Eurographics/IEEE-VGTC Symp. 2006.
- [Pfi00] Pfister, H., Lorensen, B., Bajaj, C., Kindlmann, G., Schroeder, W., Machiraju, R., "The Transfer Function Bake-Off," Proc. 11th IEEE Visualization Conf. (VIS), 523-526, 2000.
- [Rez06] Rezk Salama C., Keller M., Kohlmann P., "High-Level User Interfaces for Transfer Function Design with Semantics," IEEE Trans. Visualization and Computer Graphics, vol. 12, Issue: 5, Page(s):1021-1028, Sept./Oct. 2006.
- [Roe05] Roettger, S., Bauer, M., Stamminger, M., "Spatialized Transfer Functions," Proc. Eurographics/IEEE-VGTC Symp. Visualization (EuroVis '05), pp. 271-278, 2005.
- [Saa10] Saad A., Hamarneh G., Moller T., "Exploration and Visualization of Segmentation Uncertainty Using Shape and Appearance Prior Information," IEEE Transactions on Visualization and Computer Graphics, vol. 16, Issue:6, Page(s):1366-1375, November-December 2010.
- [Sat00] Sato Y, Westin CF, Bhalerao A, Nakajima S, Shiraga N, Tamura S, Kikinis R, "Tissue Classification Based on 3D Local Intensity Structures for Volume Rendering," IEEE Transaction Visualization and Computer Graphics, 6(2), 160-180, Apr.-June 2000.
- [Sel07] Selver M.A., Fischer F., Kuntalp M., Hillen W., "A Software Tool for Interactive Generation, Representation, and Systematical to range of Transfer Functions for 3D Medical Images," Comp. Meth. Prog. Biomed., 86, 270-280, 2007.
- [Sel08] Selver M.A., Kocaoglu A., Demir G., Dogan H., Dicle O., Guzelis C., "Patient Oriented and Robust Automatic Liver Segmentation for Pre-Evaluation of Liver Transplantation," Computers in Biology and Medicine, 38(7),:765-784, 2008.
- [Sel09] Selver M.A., Guzelis C., "Semiautomatic Transfer Function Initialization for Abdominal Visualization Using Self-Generating Hierarchical Radial Basis Function Networks" IEEE Transaction on Visualization and Computer Graphics, 15(3), 395-409, May/June 2009.
- [Shi98] Shiao-fen, F., Tom, B., Mihran, T., "Image-Based Transfer Function Design for Data Exploration in Volume Visualization," Proc. Ninth IEEE Visualization Conf., 319-326, 1998.
- [Wes10] Wesarg S., Kirschner M., Khan M. F., "2D Histogram based volume visualization: combining intensity and size of anatomical structures," International Journal of Computer Assisted Radiology and Surgery, vol. 5, Page(s): 655-666, May 2010.

Methodology for Estimation of Tissue Noise Power Spectra in Iteratively Reconstructed MDCT Data

Petr Walek
Dept. of Biomedical
Engineering
Brno University of Technology
612 00, Brno, Czech Republic
walek@feec.vutbr.cz

Jiri Jan
Dept. of Biomedical
Engineering
Brno University of Technology
612 00, Brno, Czech Republic
jan@feec.vutbr.cz

Petr Ourednicek
Dept. of Imaging Methods
University Hospital St. Anna,
Medical Fac. of Masaryk Uni.
656 91, Brno, Czech Republic
petr.ourednicek@philips.com

Jarmila Skotakova
Children's Hospital - Faculty Hospital
Masaryk University
625 00, Brno, Czech Republic
jskotakova@fnbrno.cz

Igor Jira
Children's Hospital - Faculty Hospital
Masaryk University
625 00, Brno, Czech Republic
igor.jira@fnbrno.cz

ABSTRACT

Iterative reconstruction algorithms have been recently introduced into X-ray computed tomography imaging. Enabling patient dose reduction by up to 70% without affecting image quality they deserve attention; therefore properties of noise present in iteratively reconstructed data should be examined and compared to the images reconstructed by conventionally used filtered back projection. Instead of evaluating noise in imaged phantoms or small homogeneous regions of interest in real patient data, a methodology for assessing the noise in full extent of real patient data and in diverse tissues is presented in this paper. The methodology is based on segmentation of basic tissues, subtraction of images reconstructed by different algorithms and computation of standard deviation and radial one-dimensional noise power spectra. Tissue segmentation naturally introduces errors into estimation of noise power spectra; therefore, magnitude of segmentation error is examined and is considered to be acceptable for estimation of noise power spectra in soft tissue and bones. As a result of this study it can be concluded that iDose⁴ hybrid iterative reconstruction algorithm effectively reduces noise in multidetector X-ray computed tomography (MDCT) data. The MDCT noise has naturally different characteristics in diverse tissues; thus it is object dependent and phantom studies are therefore unable to reflect its whole complexity.

Keywords

X-ray computed tomography, iterative reconstruction, dose reduction, image quality, noise power spectra.

1 INTRODUCTION

Reduction of patient radiation dose introduced by MDCT is very topical theme nowadays as a number of examinations and thus overall population radiation exposure are steadily increasing. As a result of increased effort in this branch, modern iterative reconstruction methods have been introduced, which enable reduction of patient dose by up to 70%, declaratively without affecting image quality. Such a dramatic dose reduction is enabled by inclusion of photon counting statistics nad/or models of acquisition process into the iterative

reconstruction; for further details see [BKK12] and references therein. Modern iterative reconstructions have potential to replace the conventionally used filtered back projection (FBP) algorithm; therefore, properties of iteratively reconstructed images, especially in terms of image noise, must be evaluated. Many studies dealing with problem of noise evaluation in iteratively reconstructed images have been published recently. These studies are targeted either to assessment of images acquired by scanning of artificial phantoms [MGB⁺13] or to evaluation of small regions of interest in real patient data [MNS⁺10]. According to [SS13], phantom studies cannot affect the whole complexity of noise in iteratively reconstructed images as the noise properties are dependent on imaged objects. A similar disadvantage has the latter approach, where only very limited portion of data from a specific (in many cases homogeneous) tissue is evaluated. This contribution aims to extend these analyses; a methodology for quantitative noise properties extraction from a whole

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

complex volume of iteratively reconstructed real patient data and also from fundamental tissues separately is thus presented in this paper.

2 ACQUISITION AND RECONSTRUCTION PARAMETERS

Image data processed and analyzed in this study were acquired by a Philips Brilliance 64-channel CT scanner and reconstructed by a prototype of iDose⁴ reconstructor, which uses hybrid iterative reconstruction to compute image data from a set of measured projections. A large set of one hundred and thirty real patient data has been acquired during ordinary operation of a radiological center in the University Hospital Brno - Children's Medical Center. The data set is evenly distributed between males and females aged in the range from one month to eighty years and contains images of three main body parts (head, abdominal and thoracic). Each patient was scanned either with a regular dose according to the standard scanning protocol, or in the high quality mode (HQ) or with reduced radiation dose. The rate of the dose reduction was chosen according to Tab. 1 and was controlled by respective reduction of tube anode current.

Body part	Brain	Thorax	Abdomen
	0% red.	0% red.	0% red.
Reduction	30% red. HQ	30% red.	30% red.
	30% red.	70% red.	80% red.

Table 1: Choice of dose reduction while scanning diverse body parts

Acquired raw data were reconstructed once by FBP and four times by iDose⁴, always with two differently adjusted reconstruction parameters. The two parameters were: ID level, which, according to [Hea], defines percentual strength of iterative reconstruction in reducing quantum noise, and Multi Resolution option, which can be activated additionally to any selected ID level. In this study, ID level was selected to be 30% (ID30), 50% (ID50) and 70% (ID70) independently on imaged body part, patient sex, age and dose reduction. Interpretation of the ID level is as follows: an image acquired with 30% dose reduction and reconstructed by iDose⁴ with level ID30 is declared to have equal standard deviation of noise as the identical image acquired with full dose and reconstructed using FBP.

3 TISSUES SEGMENTATION

As already stated, the main disadvantage of many studies evaluating noise properties of iteratively reconstructed images is in the fact that these approaches can not provide comprehensive and precise description of image noise when it is spatially dependent on imaged

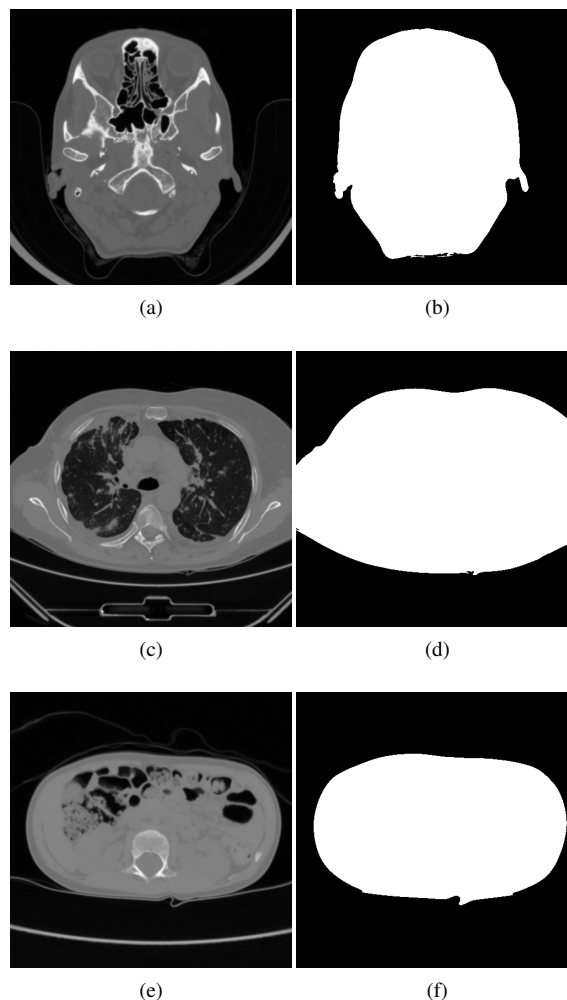


Figure 1: Examples of binary masks used for elimination of uninteresting structures computed from three main body parts: (a) slice of original brain data, (b) mask computed from (a), (c) slice of original thoracic data, (d) mask computed from (c), (e) slice of original abdominal data, (f) mask computed from (e).

anatomical structures (i.e. the nature of image noise is different e.g. in bones and in soft tissue). In order to overcome this problem, segmentation of basic tissues in acquired data must be done and noise properties must be evaluated separately in the segmented tissues.

The segmentation is based on our methodology previously published in [WJ12] where distinguishing between fundamental tissues is carried out by thresholding and subsequent classification of insufficiently segmented trabecular bones. In order to make the paper self-explaining, the methodology of the segmentation algorithm is briefly presented; however, interested readers are referred to [WJ12] for further details.

Although thresholding is conceptually very simple, a problem is in proper determination of thresholds for distinguishing between tissues, especially when

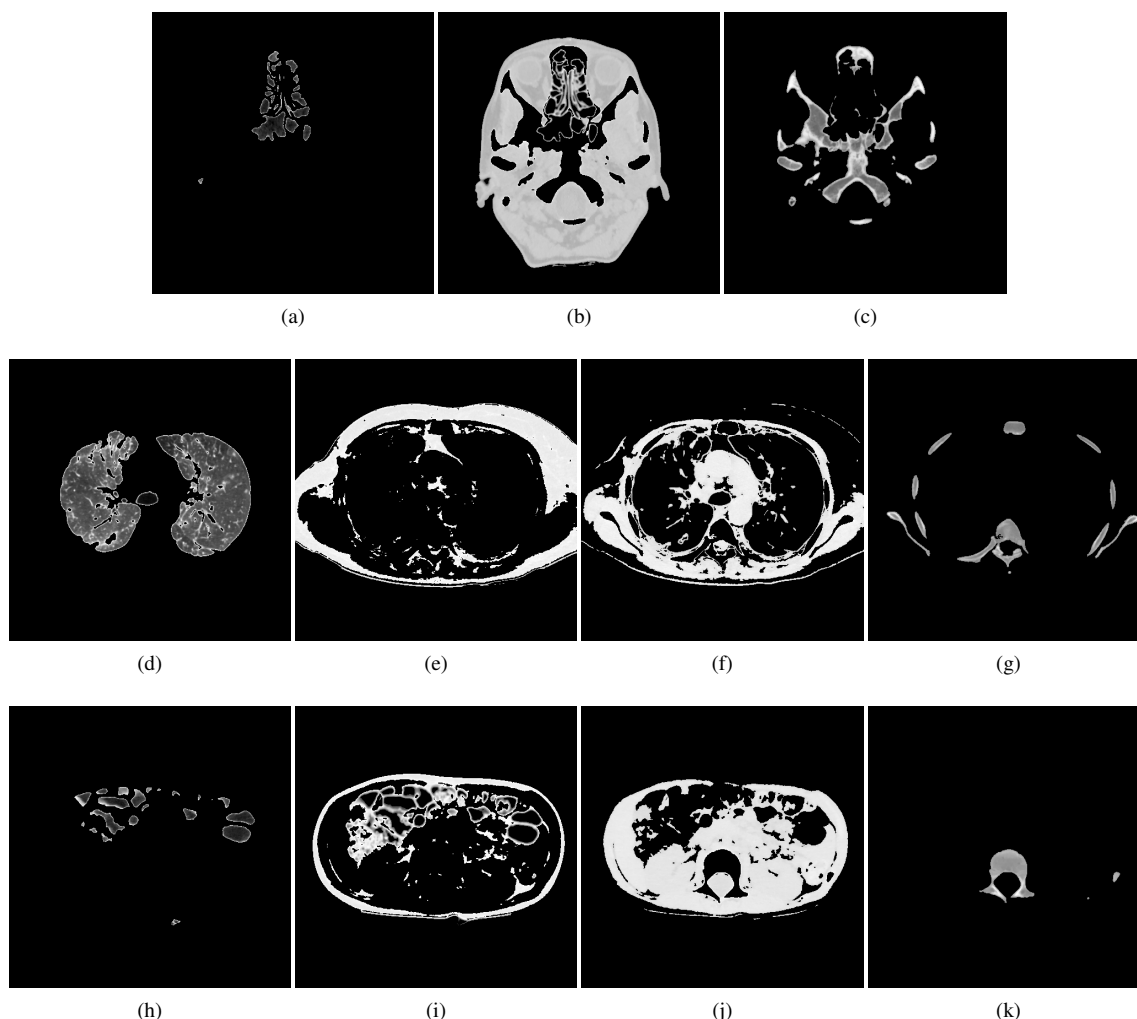


Figure 2: Images resulting from segmentation of brain 1a (a) - (c), thoracic 1c (d) - (g) and abdominal 1e (h) - (k) data: Segments of brain image are (a) paranasal sinuses, (b) soft tissue and (c) bones. Segments of thoracic image are (d) lungs, (e) adipose tissue, (f) soft tissue, (g) bones. Segments of abdominal image are (h) air in colon, (i) adipose tissue, (j) soft tissue, (k) bones.

acquired data vary substantially between imaged body parts and also within the scope of specific body part as a result of different patient age, sex, body proportions, pathologies etc. Evaluation of grayscale histogram is used to solve this problem, as histograms of the body parts are characterized by several very apparent peaks (e.g. a typical histogram of thoracic body part contains four peaks, which are representations of surrounding air, lung, adipose tissue and soft tissue). By a peak detection algorithm, one-dimensional linear and median filtering, applied consecutively on histograms, only significant peaks are detected. Positions and magnitudes of detected peaks serve as parameters of initial Gaussian curves, which are then optimally fitted to the histogram using least-squares method, and final thresholds for segmentation are subsequently determined from those optimally fitted curves.

Due to overlapping of Hounsfield units (HU) of soft tissue and inner parts of bones (trabecular bones), thresholding is unable to distinguish exactly between them. The imperfect segmentation of bones causes areas of zeros fully surrounded by ones in binary images representing segmentation of bones, which are extracted using a boundary tracking algorithm. The extracted areas represent either trabecular bones or soft tissue. Although the Hounsfield units of soft tissue and trabecular bones are partially overlapping, the shapes of their histograms differ substantially thanks to apparent texture of trabeculae in inner part of bones. In contrast with soft tissue histograms, the histograms of trabecular bones are more compact and skewed towards higher Hounsfield units. The shape differences are quantified by four parameters; entropy, compactness, skewness and kurtosis, so that each area is described by a feature vector of those parameters. The final step in segmen-

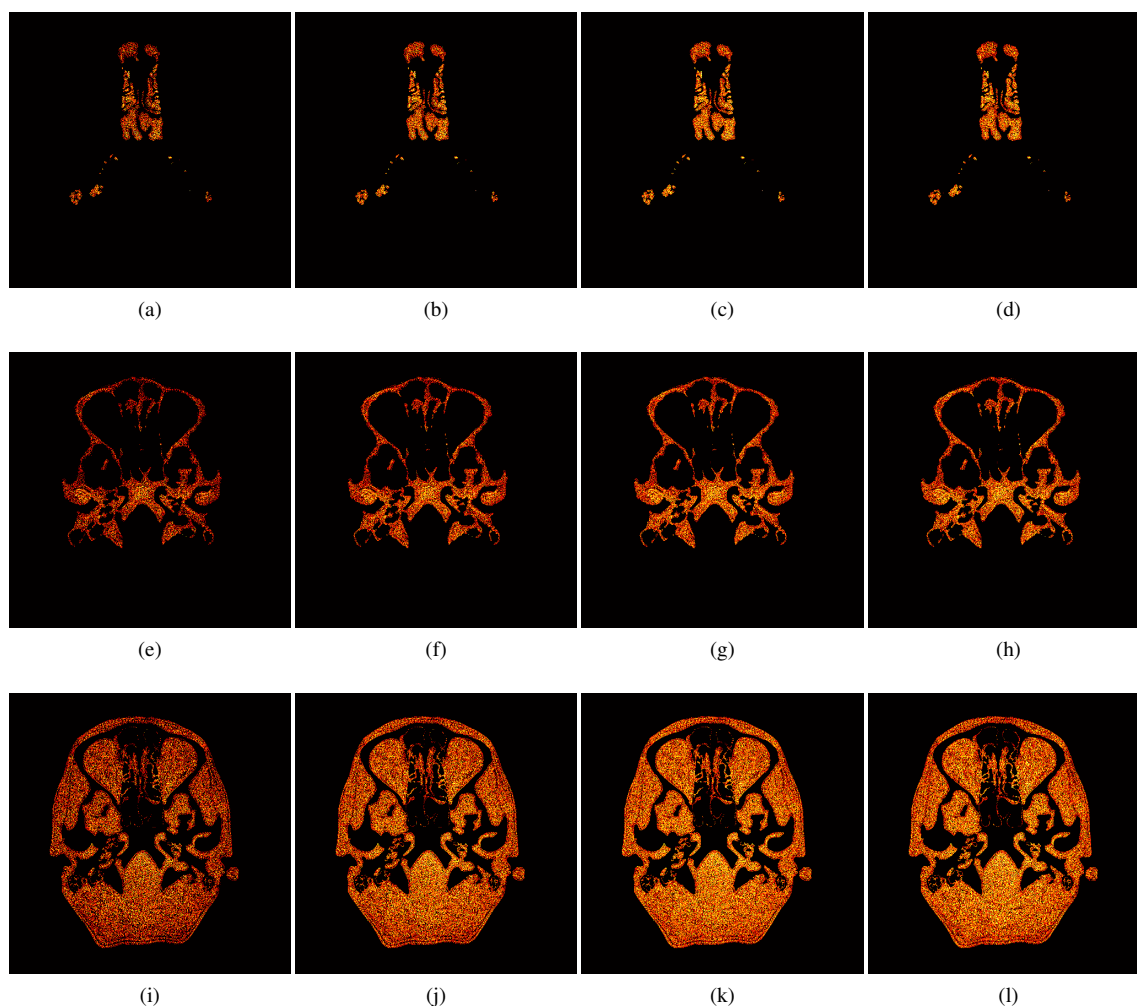


Figure 3: Residual noise images of a head body part: (a) - (d) residual noise images of paranasal sinuses, (e) - (h) residual noise images of bones, (i) - (l) residual noise images of soft tissue. First column - subtraction of ID30 and FBP reconstruction, Second column - subtraction of ID50 and FBP reconstruction, third column - subtraction of ID70 and FBP reconstruction, fourth column - subtraction of ID70MR and FBP reconstruction. The residual noise images are depicted in absolute value and logarithmic scale for highlighting small differences between them.

tation is the decision, whether extracted areas represent soft tissue or trabecular bone, based on these vectors classified by an artificial neural network.

The described segmentation algorithm is slightly modified in this paper as only noise parameters inherent to tissues of human body should be assessed. Acquired images contain also unimportant structures, such as surrounding air and patient table, which must be removed prior to applying the described segmentation. Removing of unimportant structures is performed in three simple steps. The first step is thresholding using a threshold determined by Otsu method [Ots79], which optimally distinguish between objects and background. The background involves surrounding air and low density tissues such as lung or paranasal sinuses, whereas the foreground is formed by tissues of human body and the metallic part of patient table. Connected binary seg-

ments of foreground are labeled and size of each segment is computed. Providing that the largest connected binary segment represents the human body (it is always true for our data set), other foreground structures are removed by binary area opening [Soi03]. The last step is addition of interesting structures from background (typically sinuses and lungs) to the final binary mask. With respect to continuity of human body, organs filled by air must be surrounded by tissue, and the addition can be performed by filling based on morphological reconstruction [Soi03]. Examples of binary masks used for elimination of uninteresting structures are depicted in Fig. 1 together with corresponding slices of original data. Final segmentations of basic tissues, carried out on original images depicted in Fig. 1, are presented in Fig. 2, where binary masks resulting from the segmentation are multiplied with corresponding original data.

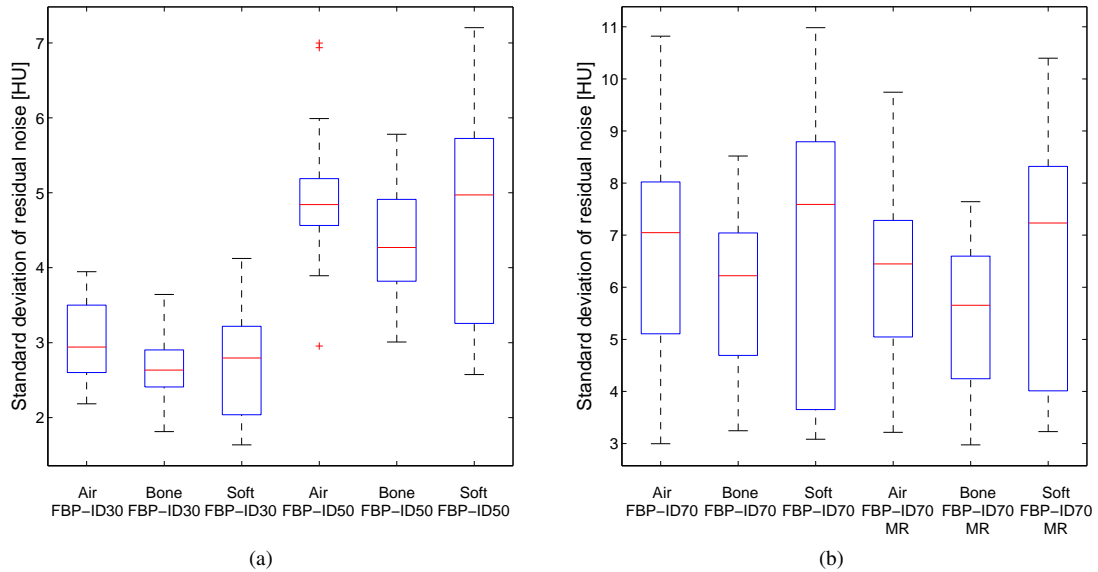


Figure 4: Box plots of residual noise standard deviations, computed from whole set of patient data, in dependence on ID level and imaged tissue. The central red line is a median, blue horizontal edges of box are 25th and 75th percentiles, dashed lines cover extent of the most extreme points and red crosses indicate outliers.

4 RESIDUAL NOISE IMAGES AND NOISE DESCRIPTORS

In order to compare noise properties of images reconstructed by iDose⁴ with images reconstructed by FBP, the anatomical structures must be removed and images of pure noise must be analyzed. Images obtained by FBP and by iDose⁴ are reconstructed from a single set of raw data (for a single patient); they are therefore in perfect spatial coherence and they can be subtracted without further registration. Providing that anatomical information is identical in every reconstruction, images of pure noise are obtained by subtraction of data reconstructed using FBP from the iteratively reconstructed data (i.e. data labeled by ID30, ID50, ID70 and ID70MR). They are called residual noise images and are subject of further analysis. According to [BCK⁺04], head images do not contain any structures likely to produce streaking artifacts; therefore their residual noise images contain only the statistical noise, which is easier to analyze. Hence, only head residual noise images will be analyzed in this paper. Binary masks resulting from the proposed segmentation are multiplied with residual noise images, see Fig. 3, so that noise properties can be evaluated in diverse tissues separately.

4.1 Standard deviation

The standard deviation (STD) provides fundamental information about the degree of statistical noise in images acquired using MDCT. Interpretation of residual noise STD is rather complicated, when taking into the account that subtraction of two random fields (i.e. noise realizations in images reconstructed by FBP and

iDose⁴) results in a new random field with a different STD. In [WJO⁺12] it is shown that STD of residual noise indicates the relative improvement of noise standard deviation in an iteratively reconstructed image with respect to the corresponding image reconstructed by FBP. Box plots, calculated from the whole set of 40 brain images, showing standard deviations of residual noise as dependent on ID level and tissue type are depicted in Fig. 4. A rising trend of residual noise STD in dependence on ID level can be observed, which according to [WJO⁺12] indicates lower content of noise in images reconstructed with higher ID level. Fig. 4 also provides evidence of spatial variance of noise as the STD is different in diverse tissues.

4.2 Radial 1D noise power spectra

Even though STD is a valuable measure of statistical noise in MDCT images, it provides information only about average magnitude of image noise; however, its frequency distribution is very important for human perception. Such information may be obtained by computation of noise power spectrum (NPS), which is often used as a quality measure of MDCT imaging systems [BCMG07], [BNCL09]. A direct digital method for computation of NPS presented in [SCJ02] and [YKH⁺08] and expressed as

$$S(f_x, f_y) = \frac{b_x b_y}{L_x L_y} \cdot \left\langle \left| DFT_{2D} \{ \mathbf{D}(x, y) - \mathbf{D}_{fit}(x, y) \} \right|^2 \right\rangle \quad (1)$$

is used. Each slice of a 3D residual noise matrix, which is considered to be one realization of a stochastic field (i.e. stochastic image), is denoted as $\mathbf{D}(x, y)$ and must

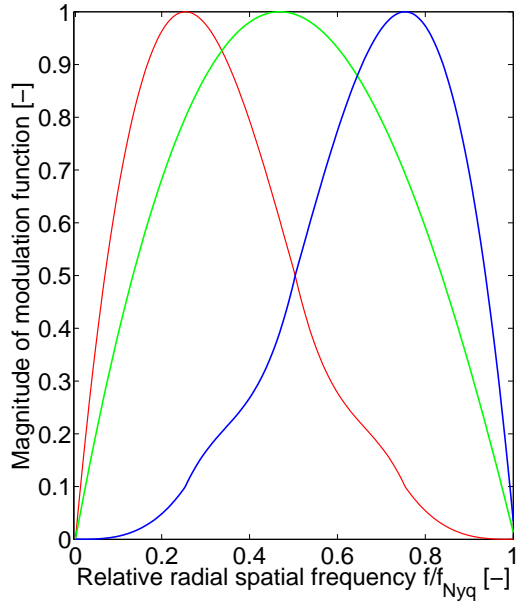


Figure 5: One dimensional functions designed for modulation of white noise spectra: (i) simulation of low frequency noise, (ii) simulation of mid frequency noise, (iii) simulation of high frequency noise.

be locally zero mean detrended prior to computation of NPS; hence subtraction of image filtered by a low pass Gaussian filter $\mathbf{D}_{\text{filt}}(x, y)$ is performed. The individual power spectrum of a noise realization is computed by squaring absolute value of 2D Fourier transform of the detrended noise realization. As individual noise power spectra suffer from large fluctuations between realizations, power spectrum of the stochastic field (i.e. the process generating random noise) must be computed as mean value of individual noise power spectra (outlined by $\langle \circ \rangle$ operator). Normalization by sampling periods b_x, b_y and image sizes L_y, L_x in directions x and y , respectively, must be done to enable comparison of power spectra of different stochastic fields. 2D NPS computed according to equation (1) is a comprehensive descriptor of noise frequency distribution; nevertheless, it is rather complicated to evaluate or extract some descriptive parameters from it. Providing that 2D NPS is rotationally symmetric, it can be expressed by radial (one-dimensional) noise power spectrum without any loss of information [BNCL09]. The radial 1D noise power spectrum is computed by angular averaging of 2D noise power spectrum and therefore is a function of the absolute spatial frequency

$$f = \sqrt{f_x^2 + f_y^2}. \quad (2)$$

Equation (1) assumes that the noise matrix $\mathbf{D}(x, y)$ is fully filled with measured noise; however, this premise is not fulfilled in case of segmented residual noise images, since they contain many zero pixels not contributing to computation of 2D NPS, see Fig. 3. Normal-

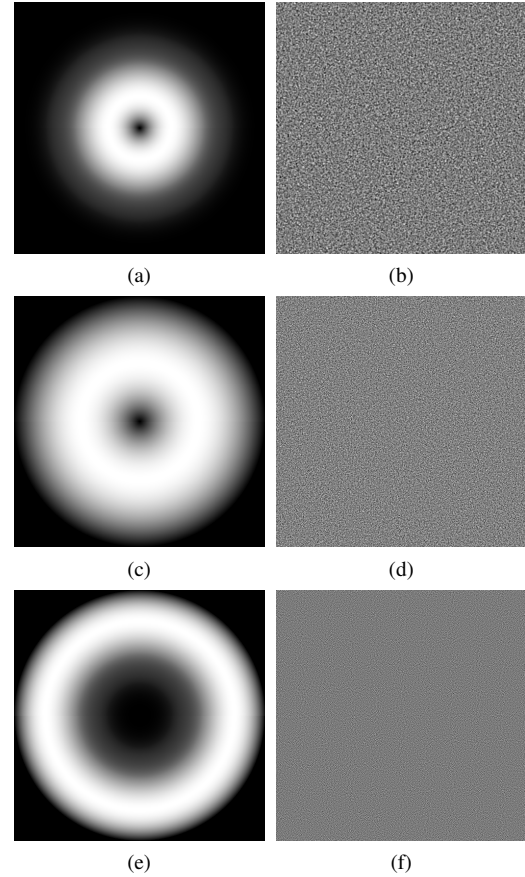


Figure 6: Two dimensional modulation functions and slices from simulated colored noise: Functions for (a) low, (c) mid and (e) high frequency noise. Examples of simulated (b) low, (d) mid and (f) high frequency noise.

ization of 2D NPS must therefore be reformulated and equation (1) is changed to

$$\mathbf{S}(f_x, f_y) = \frac{b_x b_y}{L_B} \left\langle \left| DFT_{2D} \{ \mathbf{D}(x, y) - \mathbf{D}_{\text{filt}}(x, y) \} \right|^2 \right\rangle, \quad (3)$$

where the normalization term L_B correspond to total count of pixels contributing to computation of 2D NPS. L_B can be computed as sum of pixels of the corresponding binary segmentation mask \mathbf{B} .

$$L_B = \sum_{x=1}^{L_x} \sum_{y=1}^{L_y} \mathbf{B}(x, y) \quad (4)$$

As stated before, multiplying binary masks resulting from segmentation with residual noise images is necessary; however it inevitably introduces error to estimation of NPS. According to convolutional property of Fourier transform (equation (5) [Jan06]), multiplication of two signals in the original domain corresponds to convolution of their spectra in the spectral domain.

$$FT \{ f(x, y)g(x, y) \} = \frac{1}{4\pi^2} F(f_x, f_y) * G(f_x, f_y) \quad (5)$$

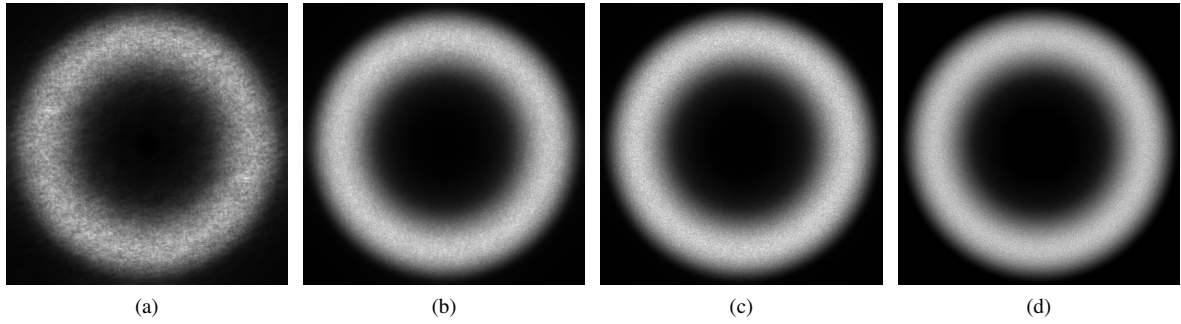


Figure 7: Two dimensional NPS computed from simulated high frequency noise multiplied with various binary segmentation masks of: (a) paranasal sinuses, (b) bones, (c) soft tissue and (d) full noise matrix.

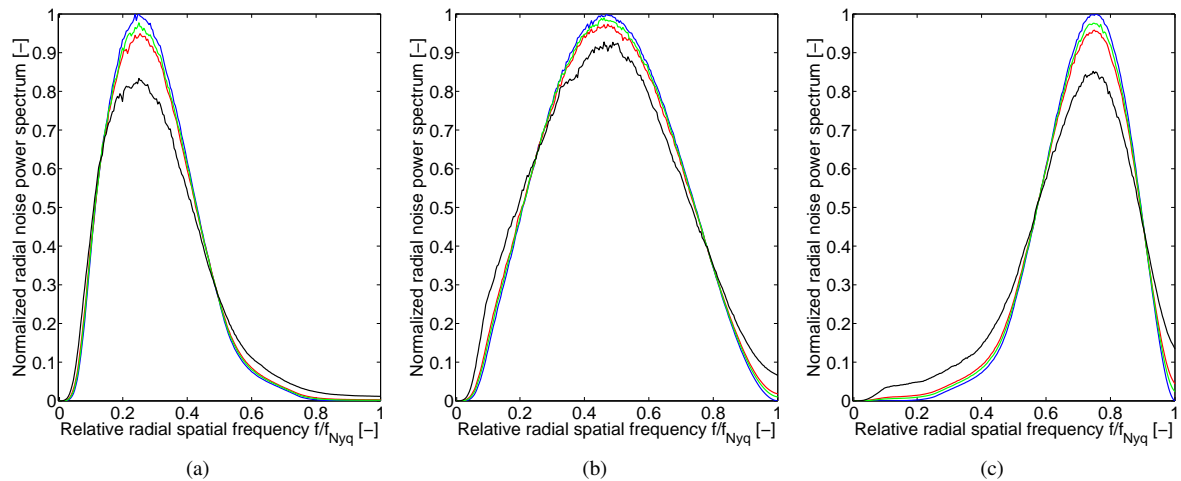


Figure 8: One dimensional NPS computed from (a) low, (b) mid and (c) high frequency noise multiplied with various binary segmentation masks of: (—) paranasal sinuses, (—) bones, (—) soft tissue and (—) full noise matrix.

The NPS of residual noise estimated by equation (3) is therefore a convolution of the sought noise power spectrum and the spectrum of a weighting binary mask, and this causes errors in estimation of 2D- and consequently of radial 1D noise power spectrum. Magnitude of the error must be evaluated prior to computation of 1D NPS of a real patient data.

4.2.1 Simulation of MDCT noise

To be able to compute errors of 1D NPS estimation caused by segmentation, three simulated matrices sized $512 \times 512 \times 512$ voxels filled with zero-mean white noise and standard deviation 10 are generated. The statistical noise inherent to MDCT images is not typically white [BCMG07]; therefore the generated model noise need to be colored (its spectral envelope must be modulated). One dimensional modulation functions based on cubic Bézier curves, see Fig.5, are designed to simulate low, mid and high frequency noise and subsequently rotationally extended to two dimensional functions MOD_{2D} . Coloring of the 3D white noise matrix is undertaken in frequency domain according to equa-

tion 6 applied slice by slice. Finally, the required 3D colored noise matrix N_{Col} is obtained,

$$N_{Col} = IDFT_{2D}\{DFT_{2D}\{N_{White}\} \cdot MOD_{2D}\}. \quad (6)$$

In Fig. 6, the 2D modulation functions can be seen together with slices taken from simulated colored noise matrices.

4.2.2 Segmentation error of radial 1D NPS

The simulated noise matrices have their noise power spectra defined by the modulation functions and thus the segmentation error of 1D NPS estimation can be evaluated. Data acquired from head body part are segmented (data reconstructed by ID70 are used as they contain the lowest portion of noise) and the three resulting binary masks (representing paranasal sinuses, bones and soft tissue) are multiplied with each of the three matrices containing low, mid and high frequency noise. Nine weighted noise matrices are together with three full noise matrices subjects of 2D- and 1D NPS computation according to equation (3). Examples of estimated 2D NPS can be seen in Fig. 7, while Fig. 8 shows 1D NPS computed from each binary mask and from low, mid and high frequency noise matrices.

	Low		Mid		High	
	Mean	Std	Mean	Std	Mean	Std
Bone	2.8	2.9	3.0	2.0	3.7	2.8
Sinuses	5.2	5.4	5.8	3.7	7.0	5.1
Soft	0.8	0.81	0.82	0.51	1.0	0.72

Table 2: Mean value and standard deviations of percentage segmentation errors.

The errors of 1D NPS estimation introduced by segmentation of tissues are computed as absolute value of differences between the estimated 1D NPS using a binary segmentation mask and the 1D NPS of full noise matrix. The errors are computed for the whole 40 patients data set and the resulting mean values and standard deviations of the errors are summarized in table 2.

The dependences of the segmentation errors on absolute spatial frequency are visualized by box plots in Fig.9, the visualization of error computed on the high frequency noise is chosen as the worst case according to table 2. The mean values and standard deviations of the errors are varying with the absolute spatial frequency. Comparing plots in Fig. 9 with the blue plot in Fig. 5, reveals that they are obviously dependent on shape of the estimated 1D NPS.

4.2.3 1D NPS of real patient data

Finally, noise power spectra of residual noise matrices in their 2D- and radial 1D forms can be computed according to equation (3) and thanks to mentioned segmentation algorithm, in each of the fundamental tissues separately. Examples of 2D NPS computed from residual noise images (FBP minus ID70) are depicted in Fig. 10 while examples of 1D NPS computed from diverse residual noise images and diverse tissues (i.e. the set of images depicted in Fig. 3) can be seen in Fig. 11.

5 CONCLUSIONS AND FUTURE WORK

A methodology for computation of radial 1D noise power spectra separately in diverse tissues of iteratively reconstructed CT image data is presented in this paper. Thanks to fully automatic segmentation algorithm and subtraction of images reconstructed by different algorithms, pure noise images representing noise inherent to fundamental tissues (called residual noise images) are obtained. Analyzing standard deviations of the residual noise images one can conclude that iterative reconstruction with higher ID level produces images with lower content of statistical noise compared to the one with lower ID level. A further deeper statistical analysis is needed to relate the improvement in suppression of image noise introduced by the iterative reconstruction to the chosen dose reduction, patient weight etc.

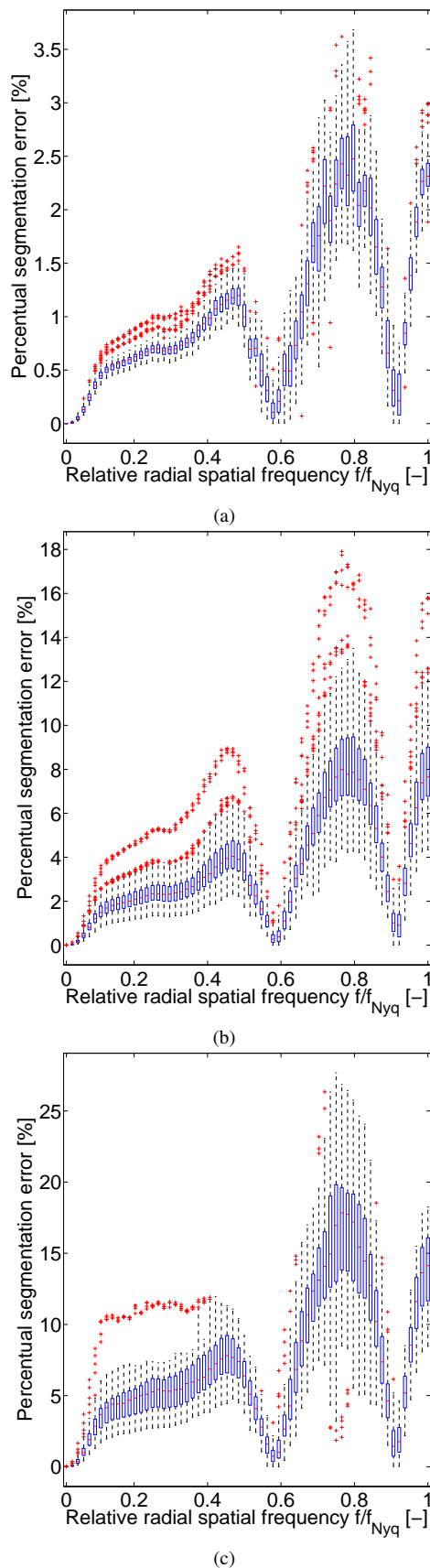


Figure 9: Box plots of errors in 1D NPS estimation introduced by segmentation of: (a) soft tissue, (b) bones, (c) paranasal sinuses.

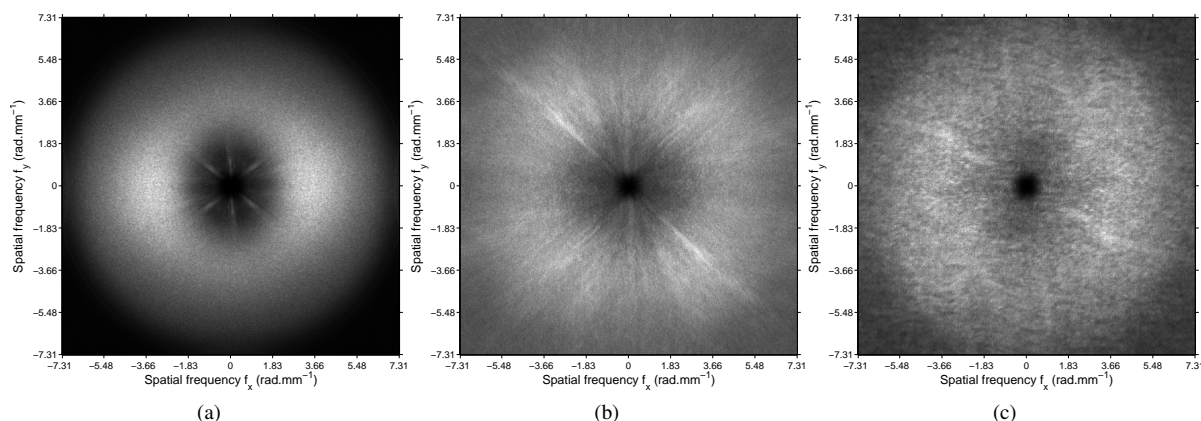


Figure 10: 2D NPS of real patient data representing frequency characteristic of noise inherent to: (a) soft tissue, (b) bones and (c) paranasal sinuses.

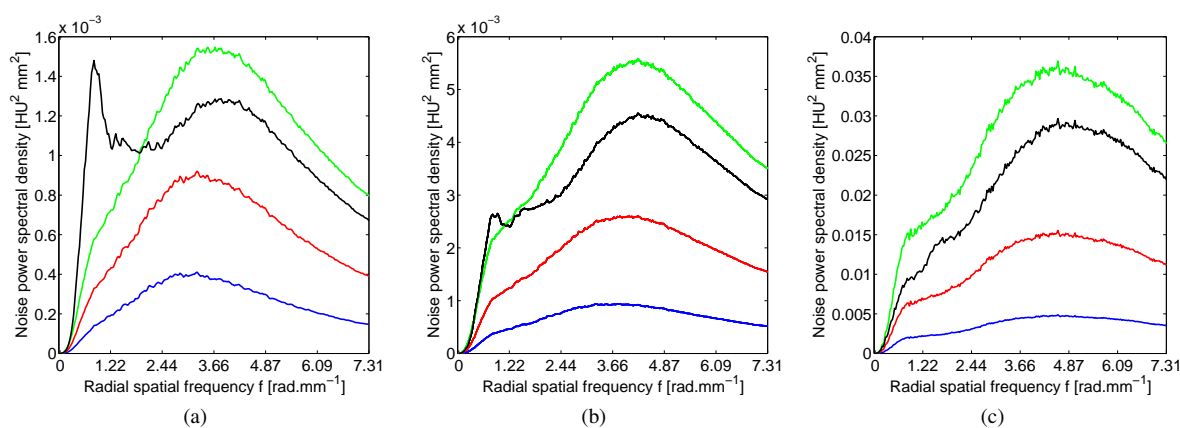


Figure 11: 1D NPS of real patient data representing frequency characteristic of noise inherent to: (a) soft tissue, (b) bones and (c) paranasal sinuses. Each plot shows four 1D NPS curves computed from diverse residual noise images: (■) FBP minus ID30, (■) FBP minus ID50, (■) FBP minus ID70 and (■) FBP minus ID70MR.

The second parameter extracted from residual noise images is radial 1D NPS. The classical equation for 1D NPS estimation is modified to fit the character of masked residual noise images as they contain many zero elements. Estimation of 1D NPS is burdened by errors as multiplying of a noise matrix with a binary segmentation mask in the original domain results in convolution of the noise spectrum with spectrum of the segmentation mask. Errors introduced by segmentation are evaluated, and it has been found that the mean value and standard deviation of errors depend on the segmented body part as well as on the frequency content of image noise, and so on the shape of estimated noise power spectra, see box plots in Fig. 9. The examined data exhibit acceptable 1D NPS estimation error in case of soft tissue (mean value of the error is under 1%) and bones (mean value of the error is under 4%). Mean value of the error of 1D NPS estimation inherent to paranasal sinuses is under 7%, and according to Fig. 9, the error can be higher even than 25%; therefore the estimation of 1D NPS inherent to paranasal sinuses produces unacceptable results.

This type of error is well known from 1D signal processing, where shortening of theoretically infinite signal (by a rectangular temporal window) causes a decrease in frequency resolution in the spectral domain as the spectrum of weighting function (sinc function) is convoluted with each discrete frequency. This effect is often referred to as spectral leakage and may be suppressed either by prolonging the weighting window or using a window with better spectral properties. Binary masks for segmenting a soft tissue have the biggest spatial extent, which explains their lowest segmentation error. As a binary weighting mask can be considered a 2D function of variable shape with steep borders, reduction of a segmentation error can be done by replacing the binary mask by a spatially variable two dimensional windowing function with better spectral properties (i.e. Hann or Hamming weighted) in our future work. A possibility for reducing the segmentation error is in careful exploration of outliers in box plots presented in Fig. 9, and in finding, which features of binary masks cause such large segmentation errors. Segmentation errors in form of small misclassified regions can represent one of

the possible reasons and its influence will be explored in our future work.

As can be seen from Fig. 10, the 2D NPS of soft tissue exhibit evident rotational asymmetry in the limited range of absolute spatial frequencies. Rotationally symmetric 2D NPS is a basic assumption to proper calculation of radial 1D NPS, hence sources of the asymmetry should be found and eliminated.

Comparing the 1D NPS depicted in Fig. 11 mutually, it can be concluded that the iDose⁴ generally preserves shape of the 1D NPS regardless of the used ID level. An exception can be seen in case of reconstruction with enabled Multi Resolution option, which exhibits a different shape of 1D NPS at low frequencies and a very high dependence on the imaged tissue. The 1D NPS also shows a dependence on the imaged scene as centroids of curves are apparently shifted along the frequency axis for different tissues.

Our future work will be focused on deriving parameters describing properties of 1D NPS inherent to diverse tissues and iDose reconstructions with varying ID level. A certain drawback of this method is the fact that extracted noise power spectra are, thanks to the subtraction, a mixture of spectral properties of iteratively- and by FBP- reconstructed images. Thus it is not clear, what does the NPS of residual noise really express. A phantom study revealing how the 1D NPS of residual noise is connected with the real noise power spectra of iteratively reconstructed image must be therefore done. The described methodology will be extended also to other body parts where problems with streaking artifacts, which can not be evaluated by NPS, must be solved. The residual noise images should be then divided into areas of statistical noise, which can be evaluated using the proposed methodology, and areas corresponding to streaking artifacts which must be addressed by a different methodology.

6 ACKNOWLEDGMENTS

Lending of the iDose⁴ reconstructor prototype software package from Philips Healthcare is highly acknowledged as well as the long term data acquisition enabled by the Clinic of Radiology, Brno Faculty Hospital - Children's Hospital.

7 REFERENCES

- [BCK⁺04] A. J. Britten, M. Crotty, H. Kiremidjian, A. Grundy, and E. J. Adam. The addition of computer simulated noise to investigate radiation dose and image quality in images with spatial correlation of statistical noise: an example application to X-ray CT of the brain. *British Journal of Radiology*, 77(916):323–328, April 2004.
- [BCMG07] K. L. Boedeker, V. N. Cooper, and M. F. McNitt-Gray. Application of the noise power spectrum in modern diagnostic MDCT: part I. Measurement of noise power spectra and noise equivalent quanta. *Physics in medicine and biology*, 52(14):4027–46, 2007.
- [BKK12] Marcel Beister, Daniel Kolditz, and Willi A. Kalender. Iterative reconstruction methods in X-ray CT. *Physica medica*, 28(2):94–108, April 2012.
- [BNCL09] Ricardo Betancourt Benítez, Ruola Ning, David Conover, and Shaohua Liu. NPS characterization and evaluation of a cone beam CT breast imaging system. *Journal of X-ray science and technology*, 17(1):17–40, January 2009.
- [Hea] Philips Healthcare. idose⁴ iterative reconstruction. *Philips Healthcare Whitepaper*. Available via https://www.healthcare.philips.com/pwc_hc/main/shared/Assets/Documents/ct/idose_white_paper_452296267841.pdf. Accessed March 2013.
- [Jan06] Jiri Jan. *Medical image processing, reconstruction and restoration: concepts and methods*. Taylor & Francis, Boca Raton, 2006.
- [MGB⁺13] Frédéric A. Miéville, François Gudinchet, Francis Brunelle, François O. Bochud, and Francis R. Verdun. Iterative reconstruction methods in two different MDCT scanners: physical metrics and 4-alternative forced-choice detectability experiments—a phantom approach. *Physica medica*, 29(1):99–110, January 2013.
- [MNS⁺10] Daniele Marin, Rendon C. Nelson, Sebastian T. Schindera, Samuel Richard, Richard S. Youngblood, Terry T. Yoshizumi, and Ehsan Samei. Low-tube-voltage, high-tube-current multidetector abdominal CT: improved image quality and decreased radiation dose with adaptive statistical iterative reconstruction algorithm—initial clinical experience. *Radiology*, 254(1):145–53, January 2010.
- [Ots79] Nobuyuki Otsu. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979.
- [SCJ02] J. H. Siewerdsen, I. A. Cunningham, and D. A. Jaffray. A framework for noise-power spectrum analysis of multidimensional images. *Medical Physics*, 29(11):2655, 2002.
- [Soi03] Pierre Soille. *Morphological image analysis*. Springer Verlag, Berlin, 2003.
- [SS13] Justin Solomon and Ehsan Samei. Are uniform phantoms sufficient to characterize the performance of iterative reconstruction in CT? In *Proc. SPIE 8668, Medical Imaging 2013: Physics of Medical Imaging*, March 2013.
- [WJ12] Petr Walek and Jiri Jan. Segmentation of basic tissues for assessing noise in iteratively reconstructed MDCT data. In *Advances in Sensors, Signals, Visualization, Imaging and Simulation*, pages 211–216. WSEAS Press, 2012.
- [WJO⁺12] Petr Walek, Jiri Jan, Petr Ourednicek, Jarmila Skotakova, and Igor Jira. Preprocessing for Quantitative Statistical Noise Analysis of MDCT Brain Images Reconstructed Using Hybrid Iterative (iDose) Algorithm. In *Journal of WSCG*, volume 20, pages 73–80, Pilsen, 2012.
- [YKH⁺08] Kai Yang, Alexander L. C. Kwan, Shih-Ying Huang, Nathan J. Packard, and John M. Boone. Noise power properties of a cone-beam CT system for breast cancer detection. *Medical Physics*, 35(12):5317, 2008.