Approximating Contact-Area of Supports in Layered Manufacturing

Ivaylo Ilinkin[*] Ravi Janardan[†] Michiel Smid[‡] Eric Johnson[†] Paul Castillo[§] Jörg Schwerdt[¶]

Abstract

This paper considers the problem of approximating the contact-area for non-convex prototypes built via Layered Manufacturing. Specifically, the paper proposes a set of heuristics for choosing candidate build directions along with a criterion to test the quality of each heuristic. The paper also presents efficient algorithms that compute approximately the amount of contact-area for a given build direction. Experimental results on real-world models provide a comparison of the proposed heuristics.

1 Introduction

Layered Manufacturing (LM) is a fast-growing technology that produces high-quality prototypes with added color in a matter of hours and at low cost. Briefly, LM works as follows: The digital description of the prototype is oriented suitably and sliced into horizontal 2D layers, which are then built in succession in the vertical direction. Ideally the prototype is oriented so that it is self-supporting during the build phase. However, real-world objects usually do not admit a self-supporting orientation, and therefore, additional structures called *supports* are created to prop up certain portions of the prototype; these are removed in post-processing.

Support requirements are measured by the volume of support structures and the extent to which supports "stick" to the prototype (the contact-area). Both depend on the orientation chosen for the prototype and must be kept small to ensure an efficient build. This problem has received considerable attention (see [1, 3, 4, 9, 13, 11, 16]). Unfortunately, few results are available for non-convex polyhedra. Majhi *et al.* [12] give support optimization algorithms for non-convex polygons in the case of 2D Stereolithography. An exact algorithm to minimize contact-area is presented in [15], but its high running time($\approx O(n^6)$) precludes its use in practice. Allen and Dutta [2] give a heuristic to approximate the minimum contact-area for non-convex polyhedra, but provide no analysis of the quality of the approximation.

In this paper we provide a suite of efficient and practical heuristics for approximating support contact-area for non-convex models and establish a ratio test that provides an indirect upper bound on the quality of the approximation of any heuristic. Our results are based on ray-shooting, convex hulls, boolean operations on polygons, spherical algorithms etc., and use the CGAL and LEDA [5, 10] libraries for implementation. Finally, we present an extensive set of experimental results on real-world STL models. Details omitted in this highly abbreviated paper (due to space limitations) may be found in [8].

2 Approximation bound

In what follows we denote by $\mathcal{P}$ the polyhedron of interest, by n the number of facets (triangles) in $\mathcal{P}$, and by $\mathbf{d}$ the *build direction*. Each facet, f, of $\mathcal{P}$ has associated with it an outward unit vector denoted by $\mathbf{n}_f$. Facet f is classified w.r.t. the given build direction $\mathbf{d}$, as *front*, *back*, or *parallel* if the angle between $\mathbf{d}$ and $\mathbf{n}_f$ is less than, greater than, or equal to $90°$, respectively. The *support contact-area* for $\mathcal{P}$ is the total surface area of $\mathcal{P}$ that is in contact with supports. Note that for a general polyhedron, while back facets are completely in contact with supports, front and parallel facets may be only partially in contact—it is this latter aspect of support structures that makes the support optimization problem that we consider so challenging.

Ideally, we would like to find a direction $\mathbf{d}^*$ that minimizes the contact-area of $\mathcal{P}$. Unfortunately, even a small change in the build orientation can result in a significantly different footprint of the supports, which makes it difficult to design a practical optimal algorithm. Therefore, an efficient heuristic that provides some guarantee about the quality of its approximation can be quite useful in practice. We develop a quality measure below.

Let $CA(\mathbf{d})$ denote the contact-area of $\mathcal{P}$ for direction $\mathbf{d}$ and let $BFA(\mathbf{d})$ be the area of all back facets with respect to $\mathbf{d}$. Let $\hat{\mathbf{d}}$ be the direction computed by a heuristic, and let $\mathbf{d}'$ be the direction that minimizes the area of back facets. Since $BFA(\mathbf{d}^*) \le CA(\mathbf{d}^*)$ and $BFA(\mathbf{d}') \le BFA(\mathbf{d}^*)$ we have,

$$\frac{CA(\hat{\mathbf{d}})}{CA(\mathbf{d}^*)} \le \frac{CA(\hat{\mathbf{d}})}{BFA(\mathbf{d}^*)} \le \frac{CA(\hat{\mathbf{d}})}{BFA(\mathbf{d}')}$$

[*]Department of Math & Computer Science, Rhodes College, ilinkin@rhodes.edu

[†]Department of Computer Science & Engineering, University of Minnesota–Twin Cities, janardan@cs.umn.edu, erj@visi.com

[‡]School of Computer Science, Carleton University, michiel@scs.carleton.ca

[§]Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, castillo17@llnl.gov

[¶]Softwarebüro Bubel GmbH, jschwerdt@swbb.de

Given any candidate direction $\hat{\mathbf{d}}$, we can use the ratio test to obtain an estimate of how close the contact-area for $\hat{\mathbf{d}}$ is to the minimum contact-area. Notice that $BFA(\mathbf{d}')$ needs to be computed only once, so the ratio test depends mainly on the efficiency of computing the contact-area for a given direction, which we describe in Section 3.

3 Contact-area on front facets

Exact algorithm: W.l.o.g. assume that $\mathcal{P}$ rests on the xy-plane and that the build direction $\mathbf{d}$ coincides with $\mathbf{z}+$. Let f be a fixed front facet and let b be any back facet. The intersection of their projections on the xy-plane yields a convex polygon, C. If the pre-images, C_f and C_b, of C on f and b, respectively, are such that C_b is above C_f in direction $\mathbf{d}$, then C_f is in contact with supports. Thus the footprint of supports on the front facet f is the union of the pre-images C_b, for all back facets b. The complexity of the union of the polygons C_b on a single front facet can be $\Theta(n^2)$. The time to compute the union is $O(n^2 \log n)$ in the worst case for any front facet [6], hence $O(n^3 \log n)$ overall. The storage requirement is $O(n^2)$; since the algorithm works on a facet-by-facet basis, the space can be reused.

A theoretically faster output-sensitive algorithm with running time $O(n \log^2 n + V \log n)$, where V is the complexity of the decomposition, is given in [17]. Unfortunately, both algorithms are extremely sensitive to degenerate input configurations, and therefore, reliable implementations require the use of exact arithmetic. However, our experiments and [17] indicate that the use of exact arithmetic introduces considerable overhead on the running time (approximately 4000 seconds for polyhedra of 2000 facets and 400 seconds for polyhedra of 150 facets, respectively). Therefore, we use an exact algorithm merely for a one-time verification of the simple heuristic below, which is practical for real data sets.

Heuristic: Let H_f (resp. M_f) denote the set of rays, originating at points on a given front facet, f, in direction $\mathbf{d}$ that hit another facet of $\mathcal{P}$ (resp. miss all facets of $\mathcal{P}$). Then the area of f that is in contact with supports can be approximated as $(|H_f|/(|H_f| + |M_f|)) * area(f)$. As the density of the sample points is increased, the accuracy of the approximation improves. The sample points are selected through a subdivision process. Each facet is subdivided progressively into triangular patches and the centroids of the patches are selected as sample points. Next each patch is subdivided into two triangles and the patches are processed in a breadth-first search fashion to ensures that all facets are subdivided to the same depth. The algorithm terminates after a predefined number of iterations or when the change in contact-area between successive iterations changes by less than 1% (each iteration processes twice as many patches as the previous one). At iteration i there are $2^i * n$ patches and for each patch the ray shooting takes $O(n)$ time. For d iterations, the running time is $O(2^d * n^2)$.

Due to the slow speed of the exact algorithm we ran comparison tests on decimated versions (2000 facets) of real-world STL models (see Table 1). (However, the final experiments in Table 1 were done on original models.) The heuristic provided nearly the same answer as the exact algorithm, but in a fraction of the time. For a decimated version of `triad` the heuristic converged to less than 1% change in contact-area in 1 second and computed contact-area of 0.32, while the exact algorithm computed 0.33 in 3888 seconds; for `top_case` the contact-area computed by the heuristic was 10494.5 in 1 second, and 10268.1 in 11592 seconds by the exact algorithm. All experiments were done on a SunBlade 100 machine with 512 MB of main memory and a 500 MHz processor. Programs were written in C++ using CGAL and LEDA [5, 10]; *Decimator*™ [7] was used to decimate the models.

A similar approach can be used to compute contact-area for parallel facets.

4 Minimizing back facet area

We map each facet, f, to a point on $\mathcal{S}^2$ (the unit-sphere) corresponding to $\mathbf{n}_f$. Let $\mathcal{C}_f$ be the set of points on $\mathcal{S}^2$ that are at distance $\pi/2$ from $\mathbf{n}_f$, i.e. $\mathcal{C}_f$ is a great circle on $\mathcal{S}^2$. (Several facets of $\mathcal{P}$ can correspond to $\mathcal{C}_f$.) Facet f will be a back facet, and therefore require supports, if and only if $\mathbf{d}$ belongs to the open hemisphere with pole $-\mathbf{n}_f$. Let $\mathcal{A}$ be the arrangement of great circles corresponding to the facets of $\mathcal{P}$.

Lemma 1 *The build direction $\mathbf{d}'$ minimizing the back facet area corresponds to a vertex in $\mathcal{A}$.*

Proof. Let c be a cell in $\mathcal{A}$. The boundary of c corresponds to front and/or back facets becoming parallel facets. Therefore, the set of back facets corresponding to any point on the boundary of c is either the same as or is a proper subset of the set of back facets corresponding to any point in the interior of c. Thus, the back facet area cannot increase. Therefore, $BFA(e) \leq BFA(c)$ for any edge e on the boundary of c.

Similarly, let e be an edge in $\mathcal{A}$ and let u be one of the vertices in $\mathcal{A}$ that is adjacent to e along the supporting great circle $\mathcal{C}_e$ of e. The vertex u represents a transition along e onto a great circle other than $\mathcal{C}_e$. Thus, a front and/or a back facet becomes parallel. As before, $BFA(u) \leq BFA(e)$. This shows that it is sufficient to examine only the directions corresponding to the vertices in $\mathcal{A}$. □

The algorithm: Lemma 1 immediately suggests an algorithm for finding $\mathbf{d}'$. We first compute the arrangement $\mathcal{A}$ of great circles on the unit sphere, pick a vertex u in $\mathcal{A}$, and initialize the back facet area term to the total area of the back facets determined by the direction corresponding to u. Next, we walk along the arrangement by visiting adjacent vertices. During the transition from vertex u to vertex v, let $\Delta BFA(u)$ be the area of the parallel facets at

u that become back facets at v, and let $\Delta BFA(v)$ be area of the parallel facets at v that were back facets at u. Then $BFA(\mathbf{d}_v) = BFA(\mathbf{d}_u) + \Delta BFA(u) - \Delta BFA(v)$. Finally, the algorithm reports the direction corresponding to the vertex for which the back facet area is minimized.

Computing the arrangement takes $O(n^2)$ time and $O(n^2)$ space. At each vertex the processing time is proportional to its degree, and therefore, the overall time for visiting the vertices is also $O(n^2)$. The space usage can be improved to $O(n)$ at the expense of increased running time to $O(n^2 \log n)$. The main idea is to focus on only a portion of $\mathcal{A}$ by walking along arc edges belonging to the same great circle. Given a great circle, $\mathcal{C}_f$, we compute its intersections with all the other great circles and sort the vertices of intersection in their circular order along $\mathcal{C}_f$. Next we pick an arbitrary vertex and initialize the back facet area term. Finally, we visit all the vertices along $\mathcal{C}_f$ and update the back facet area term. The optimal direction is identified after all great circles have been processed.

The running time per great circle is dominated by the time to sort $O(n)$ vertices of intersection in time $O(n \log n)$. The initialization time is $O(n)$ and the walk along a great circle spends per vertex time proportional to its degree, or $O(n)$ in total. Over all great circles the running time is $O(n^2 \log n)$. Since only a portion of $\mathcal{A}$ is computed, the space is $O(n)$ and this can be re-used.

5 Approximating contact-area

We present several heuristics for choosing a candidate build direction that approximates the optimal contact-area requirements. The quality of each heuristic is measured in terms of the ratio $CA(\hat{\mathbf{d}})/BFA(\mathbf{d}')$, which is an upper bound on $CA(\hat{\mathbf{d}})/CA(\mathbf{d}^*)$, as seen in Section 2. We have implemented and tested the following choices for build direction:

min BFA: Direction that minimizes the back facet area. Since the overall contact-area includes the area of the back facets, it may be advantageous to choose a direction that results in low contact-area contribution from the back facets.

max PFA: Direction that maximizes the area of parallel facets. The rationale is that parallel facets do not themselves require supports, and therefore, by maximizing their area the number of support structures could be reduced, which could lead to reduced amount of contact-area.

max PFC: Direction that maximizes the count of parallel facets. This is an alternative to the previous heuristic that maximizes the number of facets that will not require direct supports, which could lead to a reduction in support structures.

PC: Direction that corresponds to the principal components of the object (computed using MATLAB$^{\text{TM}}$ [14]). Intuitively, this heuristic builds the object along one of three mutually perpendicular axes that capture the relative shape of the object.

Flat: Direction that is opposite to the outward unit-normal of a facet of the model. It is often desirable to build the part such that it rests on one of its facets. In this case the facet must be contained in the boundary of the convex hull of the model. We select the facet on the convex hull which contains facets from the original model that have the largest total area and use as the build direction its outward unit-normal.

Random: Direction chosen at random. This heuristic was included for comparison purposes. We chose a set of one hundred and twenty random directions and computed the average of the contact-area over all of these directions, which was then used to compute the contact-area ratio. (Since the denominator in calculating the ratios is fixed, this is equivalent to taking the average of the ratios over all directions.)

Table 1 illustrates the models used for our experiment and summarizes the results. For each model the table shows the contact-area ratio computed for each heuristic and compares the ratio realized by the best heuristic with the ratio realized by the random heuristic. As seen in the column named "comp.", savings ranging from 9% to 83% are achieved on real-world (non-decimated) models. Note that even though we are comparing ratios this is equivalent to comparing the contact-areas themselves, since the denominators for both ratios are the same, namely the minimum back facet area.

6 Acknowledgement

The research of II, RJ, and EJ was supported in part by NSF grant CCR–9712226. The research of II and RJ was also sponsored, in part, by the Army HPC Research Center under the auspices of the Department of the Army, Army Research Laboratory; the content does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. The research of MS was supported by NSERC. The authors thank Stratasys, Inc. for the test data.

References

[1] P. Agarwal and P. Desikan. Approximation algorithms for layered manufacturing. In *Proc. 11th Annual ACM-SIAM SODA*, pages 528–537, 2000.

[2] S. Allen and D. Dutta. Determination and evaluation of support structures in layered manufacturing. *J. Design&Manufacturing*, 5:153–162, 1995.

[3] B. Asberg, G. Blanco, P. Bose, J. Garcia-Lopez, M. Overmars, G. Toussaint, G. Wilfong, and B. Zhu. Feasibility of design in stereolithography. *Algorithmica*, 19:61–83, 1997.

[4] P. Bose. *Geometric and computational aspects of manufacturing processes*. PhD thesis, School of Computer Science, McGill Univ., 1995.

[5] Computational Geometry Algorithms Library. `http://www.cgal.org`.

[6] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 1997.

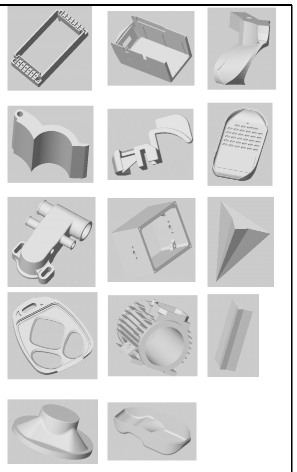
model	#facets	max ratio	min BFA	max PFA	max PFC	PC	Flat	Random	comp. (%)	Time (s)	
`oldbasex`	3660	15.60	3.33 [2]	1.72 [1]	12.40 [5]	8.16 [3]	12.40 [5]	10.37 [4]	83	270	
`tod21`	1128	7.19	1.05 [1]	1.05 [1]	3.87 [5]	3.81 [4]	1.05 [1]	4.31 [6]	76	124	
`3857438`	12184	3.37	2.63 [6]	2.54 [3]	2.54 [3]	2.41 [2]	2.32 [1]	2.55 [5]	9	3458	
`triad1`	11352	2.89	1.87 [4]	2.13 [5]	2.13 [5]	1.43 [1]	1.43 [1]	1.74 [3]	18	3262	
`ecc4`	4994	4.89	1.18 [1]	1.18 [1]	1.37 [3]	1.92 [5]	1.80 [4]	2.65 [6]	55	1272	
`cover-5`	906	5.71	3.92 [3]	3.92 [3]	3.92 [3]	3.10 [1]	4.80 [6]	3.79 [2]	18	30	
`f0m27`	3730	4.29	2.40 [4]	2.33 [1]	2.33 [1]	2.39 [3]	3.26 [6]	2.69 [5]	13	319	
`stlbin2`	2761	15.76	2.85 [2]	2.57 [1]	10.02 [5]	8.51 [3]	10.02 [5]	9.58 [4]	73	180	
`carcasse`	22876	6.23	3.77 [3]	3.47 [1]	4.19 [4]	4.38 [5]	3.47 [1]	4.89 [6]	29	12741	
`bot_case`	17642	3.04	2.11 [4]	2.11 [4]	2.11 [4]	1.54 [2]	1.29 [1]	1.95 [3]	34	7291	
`mj`	2832	5.32	2.18 [1]	2.39 [2]	2.39 [2]	2.56 [5]	2.39 [2]	3.00 [6]	27	197	
`top_case`	16692	4.09	3.14 [5]	3.14 [5]	3.07 [4]	2.14 [2]	1.97 [1]	2.50 [3]	21	6800	
`pyramid1`	10	14.34	1.00 [1]	3.65 [3]	6.02 [4]	1.00 [1]	6.02 [4]	7.27 [6]	86	5	
`prism1`	20	55.20	1.00 [1]	1.00 [1]	24.00 [4]	1.00 [1]	24.00 [4]	28.53 [6]	96	12	

Table 1: Performance of the heuristics for approximating the contact-area. The last two models were hand-generated; the remaining models are originals from Stratasys, Inc. (Models in table rows are pictured columnwise.) Running times exclude the computation of the ratio for the random heuristic, which is used only for comparison. Numbers in [ ] show the ranking of each heuristic. The column "comp." shows the fraction of improvement of the ratio given by the top-ranked heuristic over the average ratio given by the random heuristic.

[7] Decimator^{TM}1.0, Raindrop Geomagic, Inc. `http://www.geomagic.com/products/decimate/`.

[8] I. Ilinkin, R. Janardan, M. Smid, E. Johnson, P. Castillo, and J. Schwerdt. Approximating contact-area of supports in layered manufacturing. Technical Report TR–04–001, Dept. of CS&E, Univ. of Minnesota, 2004.

[9] E. Johnson. Support generation for three-dimensional layered manufacturing. Master's thesis, Dept. of CS&E, Univ. of Minnesota, 1999.

[10] Library for Efficient Data Types and Algorithms. `http://www.algorithmic-solutions.com`.

[11] J. Majhi. *Geometric methods in computer-aided design and manufacturing*. PhD thesis, Dept. of CS&E, Univ. of Minnesota, 1998.

[12] J. Majhi, R. Janardan, J. Schwerdt, M. Smid, and P. Gupta. Minimizing support structures and trapped area in two-dimensional layered manufacturing. *CGTA*, 12:241–267, 1999.

[13] J. Majhi, R. Janardan, M. Smid, and P. Gupta. On some geometric optimization problems in layered manufacturing. *CGTA*, 12:219–239, 1999.

[14] MATLAB^{TM}6.5, The Mathworks, Inc. `http://www.mathworks.com/products/matlab/`.

[15] J. Schwerdt. *Entwurf von Optimierungsalgorithmen für geometrische Probleme im Bereich Rapid Prototyping und Manufacturing*. Ph.D. thesis, Department of CS, Univ. of Magdeburg, 2001.

[16] J. Schwerdt, M. Smid, R. Janardan, E. Johnson, and J. Majhi. Protecting critical facets in layered manufacturing. *CGTA*, 16:187–210, 2000.

[17] H. Shaul and D. Halperin. Improved construction of vertical decompositions of 3d arrangements. In *Proc. 18th ACM SoCG*, pages 283–292, 2002.