# KNN Matting*

Qifeng Chen Dingzeyu Li Chi-Keung Tang
The Hong Kong University of Science and Technology

## Abstract

*We are interested in a general alpha matting approach for the simultaneous extraction of multiple image layers; each layer may have disjoint segments for material matting not limited to foreground mattes typical of natural image matting. The estimated alphas also satisfy the summation constraint. Our approach does not assume the local color-line model, does not need sophisticated sampling strategies, and generalizes well to any color or feature space in any dimensions. Our matting technique, aptly called* KNN matting*, capitalizes on the nonlocal principle by using $K$ nearest neighbors (KNN) in matching nonlocal neighborhoods, and contributes a simple and fast algorithm giving competitive results with sparse user markups. KNN matting has a closed-form solution that can leverage on the preconditioned conjugate gradient method to produce an efficient implementation. Experimental evaluation on benchmark datasets indicates that our matting results are comparable to or of higher quality than state of the art methods.*

## 1. Introduction

Alpha matting refers to the problem of decomposing an image into two layers, called foreground and background, which is a convex combination under the image compositing equation:

$$I = \alpha F + (1 - \alpha)B \tag{1}$$

where $I$ is the given pixel color, $F$ is the unknown foreground layer, $B$ is the unknown background layer, and $\alpha$ is the unknown alpha matte. This compositing equation takes a general form when there are $n \geq 2$ layers:

$$I = \sum_{i=1}^{n} \alpha_i F_i, \quad \sum_{i=1}^{n} \alpha_i = 1. \tag{2}$$

We are interested in solving the general alpha matting problem for extracting multiple image layers simultaneously with sparse user markups, where such markups may fail approaches requiring reliable color samples to work. Refer to Figures 1 and 2. While the output can be foreground/background layers exhibiting various degrees of spatial coherence as in natural image matting on single RGB images, the extracted layers with fractional alpha boundaries can also be disjoint, as those obtained in material matting from multi-channel images that capture spatially varying bidirectional distribution function (SVBRDF).

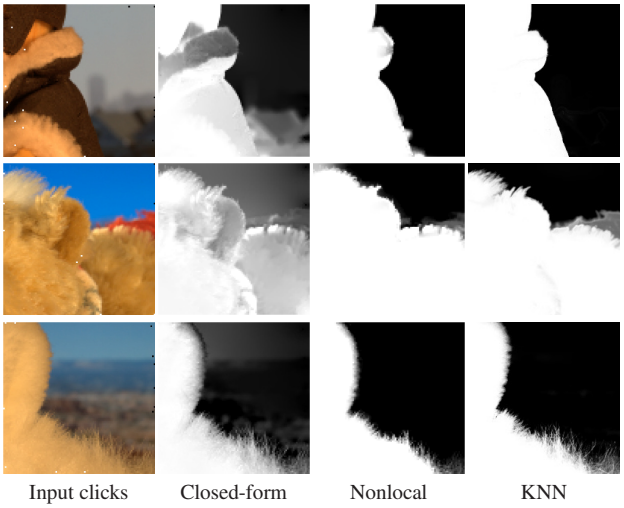


Figure 1. Using the sparse click inputs same as in nonlocal matting [11], KNN matting produces better results. Top row: clearer and cleaner boundary; middle: more details preserved for hairs as well as the red fuzzy object; bottom: furs are more clearly separated from background. Figure best viewed in electronic version.

Inspired by nonlocal matting [11], and sharing the mathematical properties of nonlocal denoising [3], our approach capitalizes on $K$ nearest neighbors (KNN) searching in the feature space for matching, and uses an improved matching metric to achieve good results with a simpler algorithm than [11]. We do not assume the *local* 4D color-line model [13, 14] widely adopted by subsequent matting approaches, thus our approach generalizes well in any color space (e.g., HSV) in any dimensions (e.g., six-dimensional SVBRDF). It does not require a large kernel to collect good samples [9, 11] in defining the Laplacian, nor does it require good foreground and background sample pairs [20, 8, 6] (which need user markups more than a few clicks, much less

*The research was supported by the Google Faculty Award and the Hong Kong Research Grant Council under grant no 619711.


Figure 2. KNN matting on material matting using the *sg* dataset. Original images at top; bottom shows sparse user input (5 clicks; one per layer) and the five layers automatically extracted. Our result distinguishes the two different gold foil layers despite their subtle difference in materials (where they were combined in [10]). Figure best viewed in electronic version.

that foreground and background are unknown themselves), and yet our approach is not at odds with these sampling approaches when regarded as postprocessing for alpha refinement akin to [8]. Our matting technique, called *KNN matting* still enjoys a closed-form solution that can harness the preconditioned conjugate gradient method (PCG) [2], and runs in order of a few seconds for high-resolution images in natural image matting after accepting very sparse user markups: our unoptimized Matlab solver runs in 30–42 seconds on an old laptop with AMD Turion X2 Ultra Dual-core Mobile ZM-80 2.1 GHz and 3G memory for images of sizes $800 \times 600$ available at the alpha matting evaluation website [1]. Experimental evaluation on this benchmark datasets indicates that our matting approach is competitive in terms of speed and quality of results.

## 2. Related Work

**Natural Image Matting** For a thorough survey on matting see [21]; here we cite the works that are closely related to ours. The matting problem is severely underconstrained with more unknowns than equations to solve, so user interaction is needed to resolve ambiguities and constrain the solution. Spatial proximity taking the form of user-supplied trimaps or strokes was employed in [4, 19], which causes significant errors when the labels are distant, and becomes impractical for matting materials with SVBRDF [12] (e.g. inhomogeneous coating by spray painting). For images with piecewise smooth regions, spatial connectivity in small image windows was used in defining the matting Laplacian [13] for foreground extraction and later in [14] for multiple layer extraction; good results are guaranteed when the linear 4D color-line model within a local window holds. Else, user needs to carefully mark up relevant colors in textured regions which are often nonlocal to one another. The closed-form solution for multiple layer extraction was analyzed in [18] where the summation and positivity constraints were investigated. The Laplacian construction and line model assumption from [13, 14] were still adopted.

The nonlocal principle has received a lot of attention for its excellent results in image and movie denoising [3]. Two recent CVPR contributions on natural image matting [11, 8] have tapped into sampling nonlocal neighborhoods. Reduced user input can be achieved by accurate *clustering* of foreground and background, where ideally the user only need to constrain a *single* pixel in each cluster for computing the optimal mattes. Thus, we prefer good clustering to good sampling of reliable foreground-background pairs for the following reasons: sampling techniques will fail in very sparse inputs that can otherwise generate good results in KNN matting; they do not generalize well to $n > 2$ layers due to the potentially prohibitive joint search space when denser input is used; adopting various modeling or sampling strategies usually leads to more complicated implementation (e.g. use of randomized patchmatch in [8], ray shooting in [6], PSF estimation in [17]), resulting in more parameter setting or requiring more careful markups/trimaps. In contrast, KNN matting requires only one non-critical parameter $K$.

The other recent CVPR contribution consists of correspondence search based on a cost function derived from the compositing equation [8]. Noting that relevant color sampling improves performance [20, 6], this approach samples and matches in a randomized manner relevant nonlocal neighbors in a joint foreground-background space which, as mentioned, can become prohibitively large if it is generalized to handle multiple layers. Earlier, a fast matting method (up to $20\times$ compared with [13]) was proposed in [9] that uses large kernels for achieving high quality results. Since the same local color-line model and the same Laplacian construction in [13, 14] were adopted, unsatisfactory results are unavoidable where large windows were used and the model assumption fails. So a separate KD-tree segmentation step was used to make the kernel size adaptive to the trimap.

**Material Matting** Much work has been done on BRDF decomposition aiming at reducing the dimensionality of a SVBRDF which is six-dimensional in its general form. Decompositions returned by principal component analysis and independent component analysis and its extensions do not in general correspond to different materials and thus are not conducive to high-level editing. Factorization approaches

input pixel (red) KNN ($10^{-5}$sec) nonlocal ($10^{-3}$sec)

Figure 3. KNN and nonlocal affinities comparison given the same pixel (marked red). Nonlocal matting uses a spatial window centered at the given pixel for sampling nonlocal neighborhoods (radius $= 9$ in [11]). KNN matting collects more matching neighborhood globally rather than within an image window, while spending significantly less computation time ($K = 81$ here).

such as homomorphic factorization [15] and matrix factorization [5] decompose a BRDF into smaller parts, but such decompositions also do not promise that individual segments correspond to single materials. Soft segmentation is required when different materials blend together. Blending weights are available in [10] where a SVBRDF was decomposed into specular and diffuse basis components which are homogeneous as previously done in [7]. In [12], a SVBRDF was separated into simpler components with opacity maps. The probabilistic formulation takes into consideration local and texture variations in their two-layer separation, and was applied successively rather than simultaneously to extract multiple material layers so accumulation errors can be resulted.

## 3. Nonlocal Principle for Alpha Matting

Akin to nonlocal matting [11] our KNN matting capitalizes on the nonlocal principle [3] in constructing affinities so that good graph clusters are obtained for respective layers. Consequently sparse input is sufficient. It was also noted in [11] that the matting Laplacian proposed in [13] in many cases is not conducive to good clustering especially when the local color-line model assumption fails, which is manifested into small and localized clusters. These clusters are combined into larger ones through a nonlinear optimization scheme in [14] biased toward binary-valued alphas.

The working assumption of the nonlocal principle [3] is that a denoised pixel $i$ is a weighted sum of the pixels with similar appearance with the weights given by a kernel function $k(i, j)$. Recall in [11] the followings:

$$E[X(i)] \approx \sum_j X(j)k(i,j)\frac{1}{\mathcal{D}_i}, \tag{3}$$

$$k(i,j) = \exp(-\frac{1}{h_1^2}\|X(i) - X(j)\|_g^2 - \frac{1}{h_2^2}d_{ij}^2) \tag{4}$$

$$\mathcal{D}_i = \sum_j k(i,j). \tag{5}$$

where $X(i)$ is a feature vector computed using the information at/around pixel $i$, and $d_{ij}$ is the pixel distance between pixels $i$ and $j$, $\|\cdot\|_g$ is a norm weighted by a center-weighted Gaussian, $h_1$ and $h_2$ are some constants found empirically.

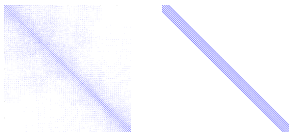

Figure 4. Typical nonlocal affinities matrix $\mathcal{A}$ in KNN matting (left, with $K = 10$) which is not as strongly diagonal as its counterpart from nonlocal matting (right, with radius $= 3$). The KNN matrix is still sparse.

By analogy of (3), the expected value of alpha matte:

$$E[\alpha_i] \approx \sum_j \alpha_j k(i,j)\frac{1}{\mathcal{D}_i} \text{ or } \mathcal{D}_i\alpha_i \approx k(i,\cdot)^T\boldsymbol{\alpha} \tag{6}$$

where $\boldsymbol{\alpha}$ is the vector of all $\alpha$ values over the input image. As described in [11]

- the nonlocal principle applies to $\boldsymbol{\alpha}$ as in (6);
- the conditional distribution $\boldsymbol{\alpha}$ given $X$ is $E[\alpha_i|X(i) = X(j)] = \alpha_j$, that is, pixels having the same appearance are expected to share the same alpha value.

The *nonlocal* principle of alpha matting basically replaces the *local* color-line assumption of [13, 14] applied in a local window which, although widely adopted, can be easily violated in practice when large kernels are used (such as [9]).

Following the derivation $\mathcal{D}\boldsymbol{\alpha} \approx \mathcal{A}\boldsymbol{\alpha}$, where $\mathcal{A} = [k(i,j)]$ is an $N \times N$ *affinity matrix* and $\mathcal{D} = \text{diag}(\mathcal{D}_i)$ is an $N \times N$ diagonal matrix, where $N$ is the total number of pixels. Thus, $(\mathcal{D} - \mathcal{A})\boldsymbol{\alpha} \approx \mathbf{0}$ or $\boldsymbol{\alpha}^T L_c \boldsymbol{\alpha} \approx 0$ where $L_c = (\mathcal{D} - \mathcal{A})^T(\mathcal{D} - \mathcal{A})$ is called the clustering Laplacian. This basically solves the quadratic minimization problem $\min_{\boldsymbol{\alpha}} \sum \mathcal{A}_{ij}(\alpha_i - \alpha_j)^2$.

In nonlocal matting the extraction Laplacian (whose derivation is more involved) rather than the above simpler clustering Laplacian was used in tandem with user-supplied input for alpha matting. While it was shown for clustering Laplacian in [11] sparse input suffices for good results, the estimated alphas along edges are not accurate due to the use of spatial patches in computing affinities $\mathcal{A}$. Moreover, the implementation in [11] requires a sufficiently large kernel for collecting and matching nonlocal neighborhoods, so specialized implementation considerations are needed to make it practical (c.f. a nice proof in fast matting [9]). The choice of empirical parameters $h_1$ and $h_2$ also affect results quality.

## 4. KNN Matting

In the following we describe and analyze our technical contributions of KNN matting which does not rely on the local color-line model, does not apply regularization, and does not have the issue of kernel size. They look straightforward at first glance (with the corresponding implementation definitely straightforward); our analysis and experimental

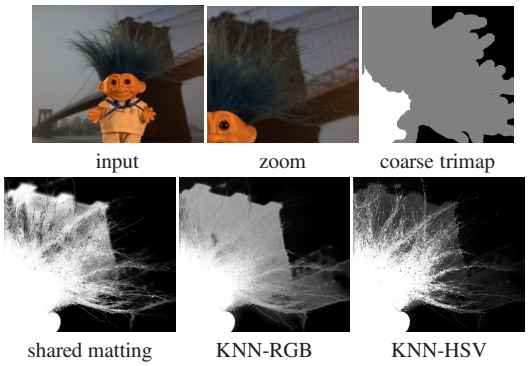


Figure 5. KNN matting can operate in any color space simply by changing the definition of feature vector in (7). Here shows the significant improvement in the result of *troll* using the HSV space on a coarse trimap. Our result is ranked top at the time of writing (followed by shared matting [6] also shown here). The hairs and the bridge are dark with close color values in the RGB space: a hair pixel has RGB $(20, 31, 33)$ and a bridge pixel $(40, 30, 33)$ in 255 scale, whereas hue of the hair is $126^\circ$ and that of bridge is $15^\circ$.

results on the other hand show that our approach provides a simple, fast and better solution than nonlocal matting [11] with generalization to multiple layers extraction. Our unoptimized Matlab implementation runs in a few seconds on $800 \times 600$ examples available at the alpha matting evaluation website [1] and our results were ranked high in [1] among the state of the arts in natural image matting. In most cases only one click is needed for extracting each material layer from SVBRDF data [10] in material matting.

### 4.1. Computing $\mathcal{A}$ using KNN

Computing $\mathcal{A}$ in KNN matting involves collecting nonlocal neighborhoods $j$ of a pixel $i$ before their feature vectors $X(\cdot)$s are matched using $k(i, j)$.

Rather than using a large kernel as in fast matting and nonlocal matting operating in the spatial image domain, given a pixel $i$, we implement the nonlocal principle by computing $K$ nearest neighbors (KNN) in the feature space. Our implementation was made easy by using FLANN [16], which proves to be very efficient in practice, running in order of a few seconds for an $800 \times 600$ image in natural image matting. We notice in nonlocal matting [11] special implementation considerations and restrictions were needed to cope with computation load with large kernels. Since kernel size is not an issue in this paper due to efficient KNN search, the running time for computing one row of $\mathcal{A}$ is $O(Kq)$ where $O(q)$ is the per-query time in FLANN. Typical values of $K$ range from only 3 (for material matting) to 15 (for natural image matting). They are not critical parameters and kept constant in our experiments.

Figure 3 compares the nonlocal neighborhoods computed using KNN matting and nonlocal matting [11], showing the efficacy of KNN searching in feature space in implementing the nonlocal principle. Figure 4 visualizes a typical $\mathcal{A}$ computed in KNN matting.

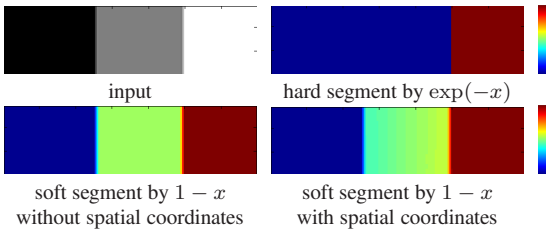


Figure 6. The $\exp(-x)$ term tends to generate hard segmentation because the kernel decreases exponentially with respect to the color difference. On the contrary, the $1 - x$ term without spatial coordinates can produce soft segmentation closer to the ground truth. Moreover, using the $1 - x$ term with spatial coordinates, we can generate an alpha matting with smoother transition between neighboring pixels.

### 4.2. Feature vector $X$ with spatial coordinates

KNN matting can be easily extended to handle SVBRDF or high dimensional data with color space other than RGB. For natural matting, a feature vector $X(i)$ at a given pixel $i$ that includes spatial coordinates to reinforce spatial coherence can be defined as

$$X(i) = (\cos(h), \sin(h), s, v, x, y)_i \tag{7}$$

where $h, s, v$ are the respective HSV coordinates and $(x, y)$ are the spatial coordinates of pixel $i$. Few previous matting approaches use the HSV color space. Feature vector can be analogously defined for material matting where a pixel has more than one observations: for material without exhibiting spatial coherence (e.g. spray paint) the spatial coordinates can be turned off.

Note the differences with nonlocal matting in enhancing spatial coherence: spatial coordinates are incorporated as part of our feature vector rather than considered separately using $d_{ij}$ in nonlocal matting (see (4)) with empirical setting of $h_2$ to control its influence. Further, image patch centered at a pixel [11] is not used in our feature vector definition. With KNN searching producing better nonlocal matching than [11], the added information of larger patch does not significantly help.

### 4.3. Kernel function $k(i, j)$ for soft segmentation

We analyze common choices of kernel function $k(x)$ to justify ours which is $1 - x$:

$$k(i, j) = 1 - \frac{||X(i) - X(j)||}{C} \tag{8}$$

where $C$ is the least upper bound of $||X(i) - X(j)||$ to make $k(i, j) \in [0, 1]$. Because (8) puts equal emphasis over the range $[0, 1]$ not biasing to either foreground or background,

the three overlapping layers can be faithfully extracted as shown in Figure 6. There is no parameter to set (c.f. $h_1$ in (4)) and KNN allows returning the smallest $||X(i)-X(j)||$.

A typical choice of kernels in machine learning, $\exp(-x)$, was used in [11]. We argue it is not a good choice for modeling optical blur and soft segmentation and in fact, it favors hard segmentation: Figure 6 shows a synthetic example where three layers are blended by fractional alphas; the same KNN matting is run on this image except that the kernel function is replaced by $\exp(-x)$. As shown in the figure hard segments are obtained. The hard segmentation results can be attributed to the non-maximal suppression property of the Gaussian kernel, where non-foreground is heavily penalized by the long tail of the Gaussian.

In nonlocal matting [11], the authors noted that the clustering Laplacian causes inaccuracy around edges, while we believe the major cause may be due to their use of the exponential term in the kernel function. Barring factors such as image outliers and color shifts due to Bayer patterns, suppose $F = (1,0,0)$ and $B = (0,0,0)$. For a pixel's value $E = (0.3,0,0)$, using (4) without the spatial term, $k(F,E) = \exp(-\|F-E\|^2/h_1^2) = \exp(-0.7^2/0.01) = \exp(-49)$ and $k(B,E) = \exp(-0.3^2/0.01) = \exp(-9)$. $k(F,E) \ll k(B,E)$ making $k(F,E)$ negligible and biasing the solution toward $B$, and thus hard segmentation is resulted. Numerically, this also causes instability in computing their clustering Laplacian, which is susceptible to singularity because many terms are negligibly small.

### 4.4. Closed-form solution with fast implementation

While the clustering Laplacian $L_c = (\mathcal{D}-\mathcal{A})^T(\mathcal{D}-\mathcal{A})$ is conducive to good graph clusters, the Laplacian $L = \mathcal{D}-\mathcal{A}$ is sparser while running much faster (up to 100 times faster than $L_c$) without compromising the results except a few more user inputs are required to achieve similar visual results. This can be regarded as a tradeoff between running time, amount of user input and result qualities. Without loss of generality $L$ is used in this section.

When user input in the form of trimaps or scribbles come along, it can be shown that the closed-form solution for extracting $n \geq 2$ layers:

$$(L+\lambda D)\sum_i^n \boldsymbol{\alpha}_i = \lambda \mathbf{m} \tag{9}$$

where $D = \text{diag}(\mathbf{m})$ and $\mathbf{m}$ is a binary vector of indices of all the marked-up pixels, and $\lambda$ is a constant controlling user's confidence on the markups. Our optimization function $g(x)$ has a closed-form solution:

$$g(x) = x^T L x + \lambda \sum_{i\in\mathbf{m}-\mathbf{v}} x_i^2 + \lambda \sum_{i\in\mathbf{v}} (x_i-1)^2 \tag{10}$$

where $\mathbf{v}$ is a binary vector of pixel indices corresponding to user markups for a given layer. Then, $g(x)$ is

$$\begin{aligned} & x^T L x + \lambda \sum_{i\in\mathbf{m}-\mathbf{v}} x_i^2 + \lambda \sum_{i\in\mathbf{v}} x_i^2 - 2\lambda \mathbf{v}^T x + \lambda|\mathbf{v}| \\ = \;& x^T L x + \lambda \sum_{i\in\mathbf{m}} x_i^2 - 2\lambda \mathbf{v}^T x + \lambda|\mathbf{v}| \\ = \;& \frac{1}{2} x^T 2(L+\lambda D)x - 2\lambda \mathbf{v}^T x + \lambda|\mathbf{v}| \\ = \;& \frac{1}{2} x^T H x - c^T x + \lambda|\mathbf{v}| \end{aligned}$$

where $\lambda|\mathbf{v}|$ is a constant. Note that $H = 2(L+\lambda D)$ is positive semi-definite because $L$ is positive semi-definite and $D$ is diagonal matrix produced by the binary vector $\mathbf{m}$. Differentiating $g(x)$ w.r.t. $x$ and equating the result to zero:

$$\frac{\partial g}{\partial x} = Hx - c = 0 \tag{11}$$

Thus the optimal solution is

$$H^{-1}c = (L+\lambda D)^{-1}(\lambda \mathbf{v}). \tag{12}$$

This echos Lemma 1 in [11] that contributes a smaller and more accurate solver than the one in [22], which gives the optimal solution in closed form.

Rather than using the coarse-to-fine technique in the solver in [13], since $H$ is a large and sparse matrix which is symmetric and semi-positive definite, we can leverage on the PCG [2] running about 5 times faster than the conventional conjugate method[1], in order of a few seconds for solving input images available at the alpha matting evaluation website. We also note that in [9] the traditional LU decomposition method and conjugate gradient method were compared. The iterative conjugate gradient method was used because for their large kernels information propagation can be faster.

### 4.5. Summation property

KNN matting in its general form for extracting $n \geq 2$ layers satisfies the summation property, that is, the estimated alphas at any given pixel sum up to 1. From (11)

$$\begin{aligned} (L+\lambda D)\boldsymbol{\alpha}_1 &= \lambda \mathbf{v}_1 \\ &\vdots \\ (L+\lambda D)\boldsymbol{\alpha}_n &= \lambda \mathbf{v}_n \end{aligned}$$

gives

$$(L+\lambda D)\sum_{i=1}^n \boldsymbol{\alpha}_i = \lambda \sum_{i=1}^n \mathbf{v}_i = \lambda \mathbf{m} \tag{13}$$

Since

$$(L+\lambda D)\mathbf{1} = \lambda D \mathbf{1} = \lambda \mathbf{m} \tag{14}$$

as the nullspace of Laplacian $L$ is $\mathbf{1}$ a constant vector with all 1's. Since $L+\lambda D$ is invertible, $\sum_{i=1}^n \boldsymbol{\alpha}_i = \mathbf{1}$.

[1] We use `ichol` provided in Matlab 2011b as the preconditioner.

In Theorem 2 of [18], the summation property was also shown for multiple layer extraction for alpha matting RGB images, where the same Laplacian from [13, 14] was still used. In practice KNN matting's output alphas are almost within $[0, 1]$. However, the summation property does not hold for sampling-based algorithms such as [8] when it comes to multiple layer extraction: to obtain the alpha matte of a layer, this layer is regarded as foreground while others as background. Consider three layers $L_1 = (1, 0, 0), L_2 = (0, 1, 0), L_3 = (0, 0, 1)$ and the pixel $I = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. To obtain the alpha matte of $L_1$, let $L_1$ be foreground $F$ and the union of $L_2$ and $L_3$ be background $B$. According to Eqn (2) in [8], $\alpha = \frac{(I-B)(F-B)}{||F-B||^2}$, the alpha value for $L_1$ is 0.5. Similarly, the alpha value for $L_2$ or $L_3$ is also 0.5. Consequently they sum up to 1.5. Normalization may help, but the normalization factor will vary from pixel to pixel. Also, the approach in [8] cannot be easily generalized to handle multiple layers due to the potentially prohibitive joint foreground-background space when multiple layers are involved.

## 5. Experimental Results

We first show in this section the results on material matting on SVBRDF data from [10] (at the time of writing the data in a very recent state of the art [12] is not available yet). Then, we will show results on natural image matting using the examples in [1], calling attention to state of the arts such as closed-form (CF) matting [13], nonlocal matting [11], fast and global matting [9, 8], and learning-based (LB) matting [22]. All of our results, including the natural image matting results and their comparisons with state-of-the-art techniques are included in the supplemental materials. Due to space limit here we highlight a few results.

### 5.1. Material matting

The clustering Laplacian was used in our material matting experiments, where a few user-supplied clicks are all KNN matting needed to produce satisfactory results shown in Figures 2 and 7. On average only one click per layer is needed. Please refer to the electronic version for best viewing of our material matting results: in *sg*, five overlapping material mattes are produced; despite that the matte for 'blue paper' has several disconnected components one click is all it takes for matting the material. KNN matting produces good mattes for *dove* where the moon and the sky mattes are soft segments, and also for *wp1* where hard segments should be produced. In *wt* the transparent slide (invisible here) was correctly matted out. In *wp2* (see supplemental material) the silver foil is brushed in three general directions, which produces different BRDF responses distinguishable in the feature space for KNN matting to output the visually correct result. In a more challenging dataset

| | | Laplacian | $n$ | max |
|---|---|---|---|---|
| *wt* | $400 \times 380 \times 162$ | 4.9 | 4 | 232.7 |
| *sg* | $500 \times 523 \times 147$ | 4.5 | 5 | 272.1 |
| *wp1* | $375 \times 480 \times 153$ | 5.52 | 2 | 153.4 |
| *wp2* | $310 \times 390 \times 153$ | 3.1 | 4 | 65.7 |
| *dove* | $510 \times 470 \times 141$ | 7.0 | 3 | 127.9 |
| *mask* | $320 \times 232 \times 93$ | 1.34 | 5 | 52 |

Table 1. Running times in secs for material matting on a machine with 3.4 GHz CPU. $n$ is number of layers; each can be computed in parallel after the Laplacian is computed. Running times shown here are the time for computing the Laplacian and the maximum time for computing an alpha layer in each example. Refer to supplemental material for other details.

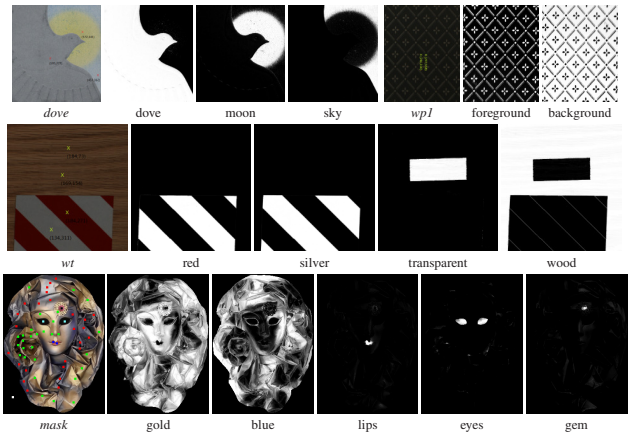


Figure 7. KNN matting on material matting. In most cases only one click per layer is needed. In *mask*, clicks with the same color belong to one layer. Figure best viewed in electronic version; see all of the material matting result images in supplemental material.

*mask*, subtle materials such as the lips and the gem were matted out. This mask example is arguably more challenging than the above for the following reasons: we use a budget capture equipment (c.f. precision equipment in [10]); the object geometry is so complex that produces a lot of cast shadows (c.f. relative flat geometry in [10]); the mixing of the blue and gold paints introduce a lot of color ambiguities. As shown in the figure more input clicks are required to produce good results. Here, spatial coordinates were not included in defining a feature vector (7) where SVBRDF does not usually exhibit strong spatial coherence. Table 1 tabulates the running times of all of the SVBRDF examples used in this paper. Thanks to FLANN computing the Laplacian takes only a few seconds for matching nonlocal neighborhoods even they are far away in the spatial domain. After computing Laplacians, individual layer extraction can be executed in parallel so we record the maximum extraction time among all layers for each example. More details are available in the supplemental material.

### 5.2. Natural image matting

The Laplacian $L = \mathcal{D} - \mathcal{A}$ was used in KNN matting in this section to obtain a sparser system for efficiency in

| | overall | avg user | pine-apple | plastic bag | normalized score(%) |
|---|---|---|---|---|---|
| Shared | 3.6 | 3.5 | 2 | 7 | 79.6 |
| Segmentation | 4.2 | 4 | 5 | 9 | 77.2 |
| **KNN** | **4.3** | **3.6** | **1** | **1** | **84.6** |
| Improved color | 4.4 | 4 | 4 | 3 | 75.7 |
| Learning-based | 5.9 | 6.4 | 12 | 2 | 67.8 |
| Closed-Form | 6 | 7.4 | 10 | 5 | 66.1 |
| Shared (real time) | 6.1 | 5.8 | 3 | 8 | 65.4 |
| Large Kernel | 6.8 | 6.5 | 6 | 4 | 62.4 |
| Robust | 7.5 | 8.1 | 8 | 6 | 55.9 |
| High-res | 8.5 | 8.1 | 9 | 13 | 51.5 |

Table 2. Overall, average and individual user-trimap MSE ranks as well as average *normalized scores* (defined in text) on the alpha matting evaluation website [1]. Running time of our codes can be tenfold faster by employing GPU implementattion of KNN and PCG. Complete ranking and detailed information on GPU acceleration are in supplemental material.

our natural image matting experiments. Table 2 tabulates the partial ranking among the methods evaluated in [1], showing that KNN matting is competitive overall on the same dense trimaps without sophisticated sampling strategies. Figure 8 shows the qualitative comparison of selected examples on fuzzy objects and objects with holes (with complete results and comparison with CF and LB matting in [1] available in the supplemental material), noting the *pineapple* used in [9] as a failure case on local color-line assumption [13] whereas KNN matting performed better than shared matting on this example (Table 2) without sophisticated sampling strategies.

KNN matting gives top performance on difficult images (*plastic bag* and *pineapple*, Figure 8) while [1] does not rank us high on arguably easier ones (*donkey* and *elephant*, see supplemental material), although we obtain good alpha mattes quantitatively the same as other top-ranked methods on such easier examples. For this reason, we define the *normalized score* of a method given a trimap as the ratio of the best MSE for that trimap to its MSE. We argue that normalized scores are fairer than average ranks: for the *donkey* user-trimap, the top 10 methods have the *same* MSE 0.3, but shared matting ranks first while robust matting ranks 10th. More alternative ranking information can also be found in the supplemental material. In summary, regardless of ranking methods, given the trimaps from [1], our results are better than closed-form matting [13], fast and global matting [9, 8], and visually similar to the high-quality results of shared matting [6].

At times a lay user may not be able provide detailed trimaps akin to those in [1]; a few clicks or thin strokes are expected. Figure 1 shows our visually-better results compared with nonlocal matting [11] based on the same input clicks used in their paper. Figure 9 compares the results on very sparse input, showing that KNN matting preserves better the fuzzy boundaries as well as the solid portions of the foreground than state-of-the-arts. Figure 10 shows the MSE

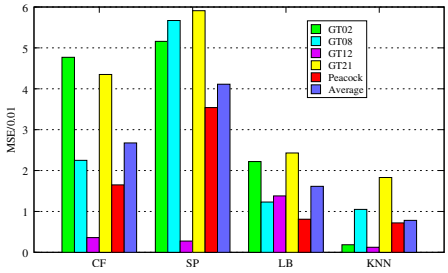


Figure 10. For very sparse input in Figure 9, KNN matting is better than other state-of-the-art matting approaches.

comparison of our method with closed-form matting, spectral matting, learning-based matting on six examples with ground-truth, where the input consists of only a few strokes.

## 6. Conclusion

Rather than adopting the color-line model assumption in a local window or relying on sophisticated sampling strategies on foreground and background pixels, we propose KNN matting which employs the nonlocal principle for natural image matting and material matting, taking a significant step to produce a fast system that outputs better results and is easier to implement (our implementation has only about 50 lines of Matlab codes). It generalizes well to extracting $n \geq 2$ multiple layers in non-RGB color space in any dimensions where kernel size is also not an issue. Our general alpha matting approach allows the simultaneous extraction of multiple overlapping layers based on sparse input trimaps and outputs alphas satisfying the summation constraint. Extensive experiments and comparisons using standard datasets show that our method is competitive among the state-of-the-arts. Meanwhile, because KNN matting constructs clustering Laplacian based on feature vector, the choice of elements in feature vector is very important. As shown in Figure 5, KNN matting is better on HSV color space rather than RGB color space. Thus inappropriate feature vector can lead to undesirable results. Future work includes investigating the relationship between the nonlocal principle and the color-line model applied *nonlocally* in general alpha matting of multiple layers from images.

## References


[1] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott. *A perceptually motivated online benchmark for image matting*. In *CVPR*, 2009.

[2] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[3] A. Buades, B. Coll, and J.-M. Morel. Nonlocal image and movie denoising. *IJCV*, 76(2):123–139, 2008.

[4] Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski. A bayesian approach to digital matting. *CVPR'01, II:264-271*, 2001.

[5] F. H. Cole. Automatic brdf factorization. Bachelor Honors Thesis, Harvard College, Cambridge, MA, 2002.

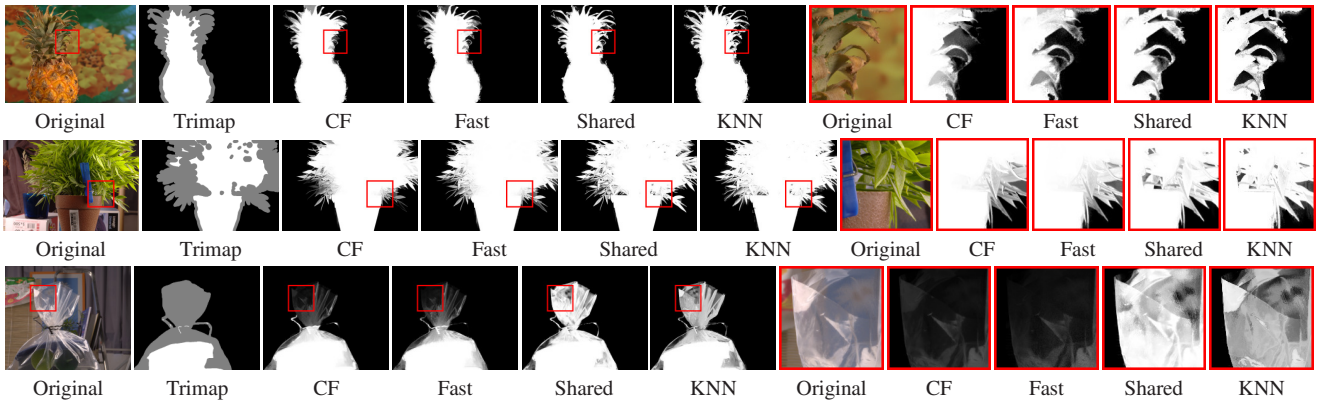


Figure 8. KNN matting on natural image matting. The MSE rankings are from [1]. Top: images with very similar background/foreground color and fuzzy boundary; KNN ranks first. Middle: Images with holes; KNN ranks third. Bottom: Images with high transparency; note our result is more reasonable because the alphas should be around 0.5 rather than 0 or 1 for the transparent nature; KNN ranks first in this example. Figure best viewed in electronic version. See more comparisons in supplemental material.

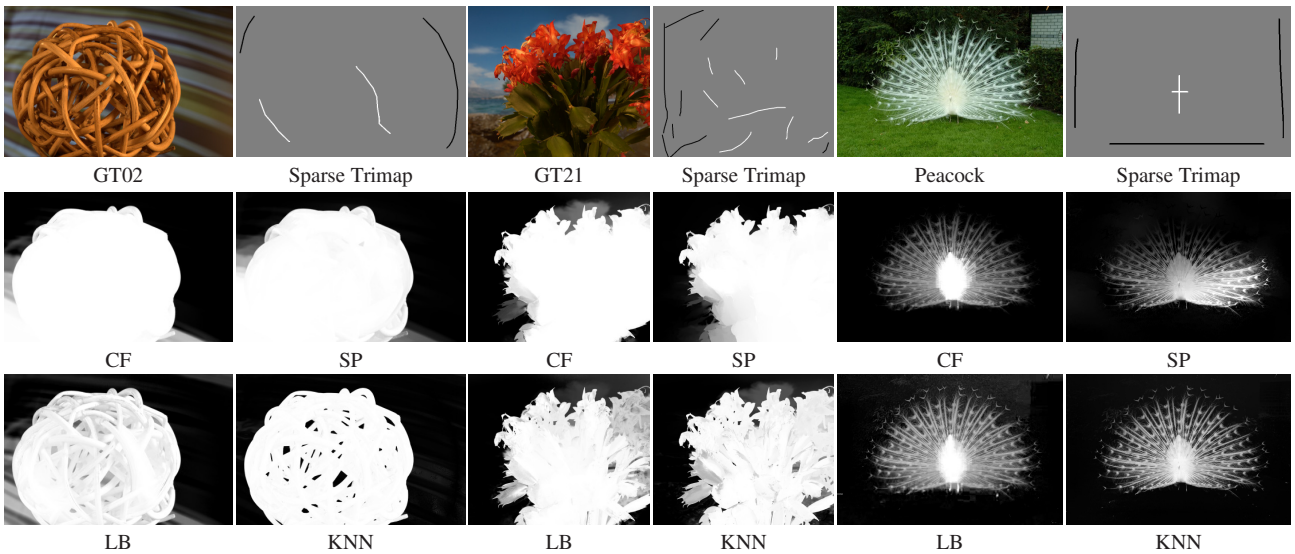


Figure 9. Comparison on sparse user-supplied trimaps. KNN matting produces better results in around 30 seconds using PCG in each case whereas it takes 300 seconds for SP matting. See more comparisons in supplemental materials. Figure best viewed in electronic version.


[6] E. S. L. Gastal and M. M. Oliveira. Shared sampling for real-time alpha matting. *Computer Graphics Forum*, 29(2):575–584, May 2010.

[7] D. B. Goldman, B. Curless, A. Hertzmann, and S. M. Seitz. Shape and spatially-varying brdfs from photometric stereo. *IEEE TPAMI*, 32(6):1060–1071, 2010.

[8] K. He, C. Rhemann, C. Rother, X. Tang, and J. Sun. A global sampling method for alpha matting. In *CVPR'11*, pages 2049–2056, 2011.

[9] K. He, J. Sun, and X. Tang. Fast matting using large kernel matting laplacian matrices. In *CVPR*, pages 2165–2172, 2010.

[10] J. Lawrence, A. Ben-artzi, C. Decoro, W. Matusik, H. Pfister, R. Ramamoorthi, and S. Rusinkiewicz. Inverse shade trees for non-parametric material representation and editing. *ACM Transactions on Graphics*, pages 735–745, 2006.

[11] P. Lee and Y. Wu. Nonlocal matting. In *CVPR*, pages 2193–2200, 2011.

[12] D. Lepage and J. Lawrence. Material matting. *ACM Trans. Graph.*, to appear.

[13] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *IEEE TPAMI*, 30(2):228–242, 2008.

[14] A. Levin, A. Rav-Acha, and D. Lischinski. Spectral matting. *IEEE TPAMI*, 30:1699–1712, October 2008.

[15] M. D. McCool, J. Ang, and A. Ahmad. Homomorphic factorization of brdfs for high-performance rendering. In *SIGGRAPH '01*, pages 171–178, 2001.

[16] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISSAPP'09*, pages 331–340, 2009.

[17] C. Rhemann, C. Rother, P. Kohli, and M. Gelautz. A spatially varying psf-based prior for alpha matting. In *CVPR*, pages 2149–2156, 2010.

[18] D. Singaraju and R. Vidal. Estimation of alpha mattes for multiple image layers. *IEEE TPAMI*, 33(7):1295–1309, July 2011.

[19] J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum. Poisson matting. *ACM Trans. Graph.*, 23:315–321, August 2004.

[20] J. Wang and M. F. Cohen. Optimized color sampling for robust matting. *CVPR '07*, 2007.

[21] J. Wang and M. F. Cohen. *Image and Video Matting*. Now Publishers Inc., Hanover, MA, USA, 2008.

[22] Y. Zheng and C. Kambhamettu. Learning based digital matting. In *ICCV*, pages 889–896, 2009.