# Nonlocal Matting

Philip Lee and Ying Wu
Northwestern University
2145 Sheridan Road, Tech L359
Evanston, IL 60208
{pgl622,yingwu}@eecs.northwestern.edu

## Abstract

*This work attempts to considerably reduce the amount of user effort in the natural image matting problem. The key observation is that the nonlocal principle, introduced to denoise images, can be successfully applied to the alpha matte to obtain sparsity in matte representation, and therefore dramatically reduce the number of pixels a user needs to manually label. We show how to avoid making the user provide redundant and unnecessary input, develop a method for clustering the image pixels for the user to label, and a method to perform high-quality matte extraction. We show that this algorithm is therefore faster, easier, and higher quality than state of the art methods.*

## 1. Introduction

Natural image matting is the problem where an image is to be decomposed into two layers, called foreground and background, which are blended together linearly with an alpha mask as follows:

$$I_i = \alpha_i F_i + (1 - \alpha_i) B_i, \tag{1}$$

where $I$ is the given image, $F$ is the unknown foreground layer, $B$ is the unknown background layer, $\alpha$ is the unknown alpha matte, and $i$ is the pixel index. Since there are three unknowns at each pixel $i$ with only one constraint, the problem is severely underconstrained.

In order to solve this problem, it is necessary that a human provide enough input to resolve the ambiguities and constrain the solution. Typically, this input can be a trimap or scribbles, which are effectively the same. Some methods [18, 3, 6] require densely populated trimaps. The reason they need detailed input is because they all construct the matte by spatial proximity to the nearest user constraint. But due to this, they accumulate significant errors when the constraint is distant. [10, 11, 19, 1, 15, 5] are successful on certain sparser input, due to an explicit or implicit sparsity assumption. For example, [10] models the pixel distribution as two lines in RGB space, and therefore gains

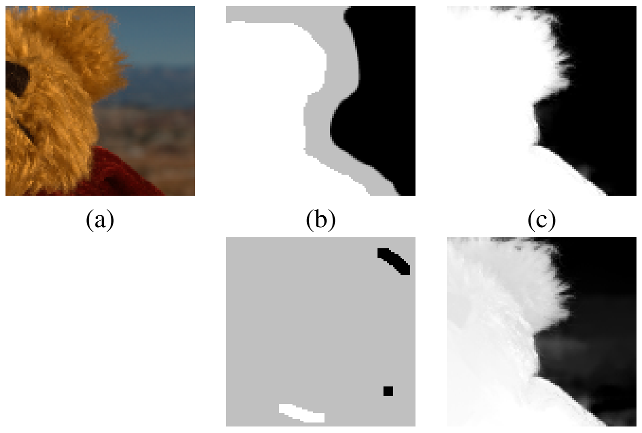


Figure 1. (a) Input image. (b) Dense trimap (white regions indicate foreground constraints, black indicates background constraints, and gray indicates no constraint). (c) [10] on dense input. (d) Sparse trimap. (e) [10] on sparse input. Current methods like [10] fail on sparse input. We remedy this.

accuracy when the assumptions hold. [15] utilizes an edge-preserving sparsity prior directly on the matte. In spite of this, these methods have to employ a level of spatial regularization, preventing them from taking the next step to even sparser input.

Other fundamentally different approaches to matting exist that reduce the user input to almost zero. [12, 14] use focus blur to extract depth information to classify the objects into foreground or background. [9] does the same by creating a synthetic aperture from a camera array, and [20] directly uses depth information from a depth camera. But here, we focus on the case that we have a simple camera and no such sophisticated equipment.

User input plays an important role in obtaining a quality matte. In general, the more input, the better the solution. Of course, this requires more human effort. Scribble-based input is simple and is generally good for smooth image re-

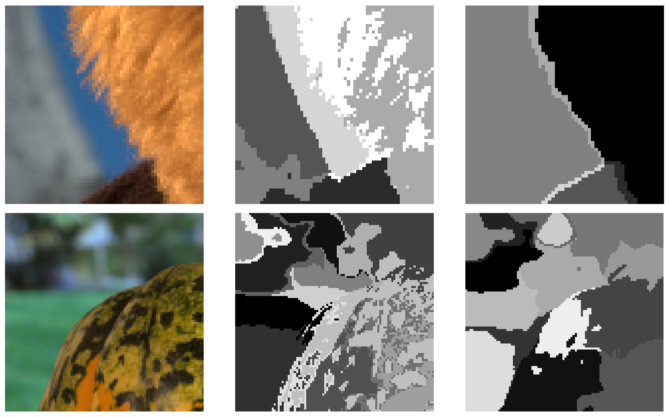


Figure 2. (a) Input image. (b) Labels after clustering Levin's Laplacian. (c) Clustering nonlocal Laplacian.

gions, but we observe they are often ineffective for textured regions unless the user makes painstakingly detailed scribbles over the region. So, a very interesting question is: how can we minimize human effort in solving the matting problem?

To the best of our knowledge, there is very little in the literature to answer: what is good input, and how can we provide the least amount of input for the most accuracy? We answer these questions by showing that the nonlocal principle [2] acts as a very effective sparsity prior on the matte, and serves to *dramatically* reduce the human effort required to generate an accurate matte. Using this technique, we can effectively handle textured regions, edges, and structure in a way that no one has demonstrated before.

## 2. What is Good Input?

So far, most papers on natural image matting have ignored one major question: what is good user input? In other words, what is the best input a user can provide so that the algorithm gives the most accurate results? This question can be difficult to answer, but viewing the matting Laplacian as a graph Laplacian can give some insights.

Graph Laplacians have been popularized in matting literature and other applications for their role in segmentation and embedding [5]. If there are $n$ points to label, they define an affinity $A_{ij}$ between points $i$ and $j$, and store all affinity pairs in matrix $A$. The Laplacian is given as $L = D - A$, where $D$ is diagonal and $D_{ii} = \sum_j A_{ij}$. Then, labeling all the data can be posed as a constrained minimization, where the objective function is $x^T Lx$. This is a nice, convex, quadratic program that can be easily solved. The key factor in performance is the design of the affinities $A_{ij}$.

In [10], the pixel affinity is defined as follows. They assume that in a small spatial patch of each pixel $i$, the image is composed of convex combinations of colors from a foreground color line and background color line. When this assumption holds, as well as a few other technical conditions, a linear regresser can exactly recover the alpha matte. Denote by $\boldsymbol{\alpha}_i$ the collection of matte values in patch $i$, i.e. $\boldsymbol{\alpha}_i = [\alpha(\tau_1), \ldots, \alpha(\tau_m)]^T$. All the pixels in the patch are collected in a data matrix:

$$\mathbf{X}_i = \begin{bmatrix} X^R(\tau_1) & X^G(\tau_1) & X^B(\tau_1) & 1 \\ & \vdots & & \\ X^R(\tau_m) & X^G(\tau_m) & X^B(\tau_m) & 1 \end{bmatrix}, \qquad (2)$$

where $\{\tau_1, \ldots, \tau_m\}$ is a patch of pixels around pixel $i$. So, the local affinity matrix in [10] is defined as

$$A_i = \mathbf{X}_i \mathbf{X}_i^\dagger, \qquad (3)$$

where $\mathbf{X}_i^\dagger$ is the pseudo-inverse of the data matrix. Basically, this affinity matrix tells how often two pixels are classified the same by the linear classifier in patch $i$. The normal vector for the classifier's decision boundary is given by $\mathbf{X}_i^\dagger \boldsymbol{\alpha}_i$. The total affinity matrix is calculated by summing the $A_i$ together.

Matting (or classification) Laplacians have a direct interpretation as a graph Laplacian, where each pixel or data point is a node in a graph $G = (V, E)$, and $A_{ij}$ represents the weight of edge $(i, j) \in E$. The quadratic form $q(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^T L \boldsymbol{\alpha}$ has the particularly nice form $q(\boldsymbol{\alpha}) = \sum_{(i,j)\in E} A_{ij}(\alpha_i - \alpha_j)^2$, which is a measure of smoothness along the edges of $G$. It is directly obvious from $L$ that the vector of all ones $\mathbf{1}$ is an eigenvector of $L$ having eigenvalue 0. However, the rest of the eigenvectors are far from trivial, and play crucial roles in determining the nature of $G$.

Suppose there is a subset of pixels that exactly cluster in the graph implied by Laplacian $L$, and that we constrain one of the pixels in that cluster to have alpha value $\overline{a}$. Then, the value of the objective function $q(\boldsymbol{\alpha})$ is minimized if the rest of the pixels in the cluster are also labeled $\overline{a}$. The reason is that, if we have $K$ clusters, $q(\boldsymbol{\alpha})$ can be decomposed as

$$q(\boldsymbol{\alpha}) = \sum_{k=1}^{K} \sum_{(i,j)\in E_k} A_{ij}(\alpha_i - \alpha_j)^2, \tag{4}$$

where $\{E_k \mid k = 1, 2, \ldots, K\}$ is a partition of $E$. Therefore, labelling the rest of the values in the cluster as $\overline{a}$ minimizes $q(\boldsymbol{\alpha})$. What this means is that if we can construct the affinities so that the graph clusters accurately on foreground and background objects, the user only needs to constrain a single pixel in each cluster, and the algorithm can do the rest of the work.

So, it is enough for the user to label a single pixel in each of the clusters of $L$. Therefore, the task becomes how to construct $L$ in a way that maximizes the size of the clusters while minimizing their total number so that the user's job is easy.

## 3. Reducing User Input

In many cases, obtaining a good clustering is difficult by using Levin *et al.*'s Laplacian [10]. The main reason is that there is an overly strong regularization of the problem with spatial patches. This causes many clusters to be small and localized (Fig. 2.b). This may be fine for large smooth regions, but is inappropriate for textured regions. In an attempt to remedy this, Levin *et al.* presented another algorithm in [11] that combined the small clusters into larger ones that they call *matting components* based on two priors: the components are heavily biased to be binary-valued, and they do not overlap. Their solution is a nonlinear optimization procedure with a non-convex and non-differentiable objective function.

In this paper, we present a new approach that achieves better results *directly*. This new approach defines a new Laplacian based on the nonlocal principle. The nonlocal principle [2] essentially states that given an image $X$, the denoised pixel $i$ is a weighted sum of the pixels that have similar appearance, where the weights are given by a kernel $k(i, j)$. More formally,

$$E\left[X(i)\right] \approx \sum_j X(j)k(i,j)\frac{1}{D_i}, \text{ where} \tag{5}$$

$$k(i,j) \triangleq \exp\left(-\frac{1}{h_1^2}\left\|X(S(i)) - X(S(j))\right\|_g^2 - \frac{1}{h_2^2}d^2(i,j)\right) \tag{6}$$

$$D_i \triangleq \sum_j k(i,j). \tag{7}$$

Here, $S(i)$ is a small *spatial patch* around pixel $i$, and $d(i,j)$ is the pixel distance between pixels $i$ and $j$. $\|\cdot\|_g$ indicates

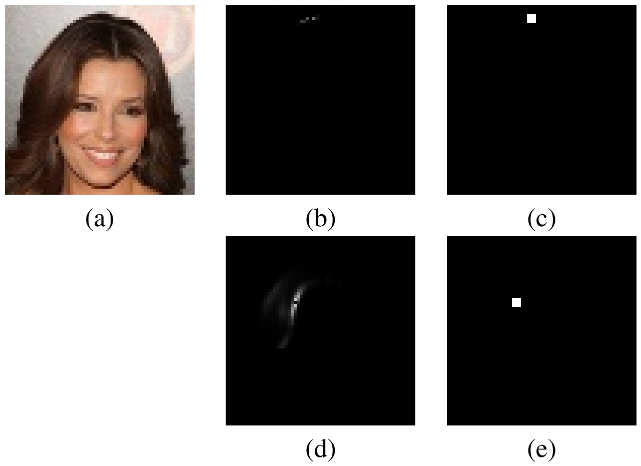


Figure 3. (a) Original. (b) Nonlocal neighbors at pixel (5,29). (c) Spatial neighbors at pixel (5,29). (d) Nonlocal neighbors at (24,23). (e) Spatial neighbors at (24,23).

that the norm is weighted by point-spread function (PSF) $g$, which we take to be a center-weighted Gaussian PSF with standard deviation on the order of the radius of spatial neighborhood $S(i)$. See Fig. 3 for a visualization of $k(i,\cdot)$ on a typical image, and note how it is fundamentally different from the $3\times 3$ spatial connectivity of Levin *et al.* [10, 11].

Essentially, the nonlocal principle defines a discrete data distribution $k(i,\cdot)/D_i$ for each pixel $i$, and implies that the image can be sparsely represented by a few spatial patches. We take advantage of these facts and extend them to the alpha matte.

Suppose that, by analogy to (5), the alpha matte is such that

$$E\left[\alpha_i\right] \approx \sum_j \alpha_j k(i,j)\frac{1}{D_i}, \tag{8}$$

or, we can write

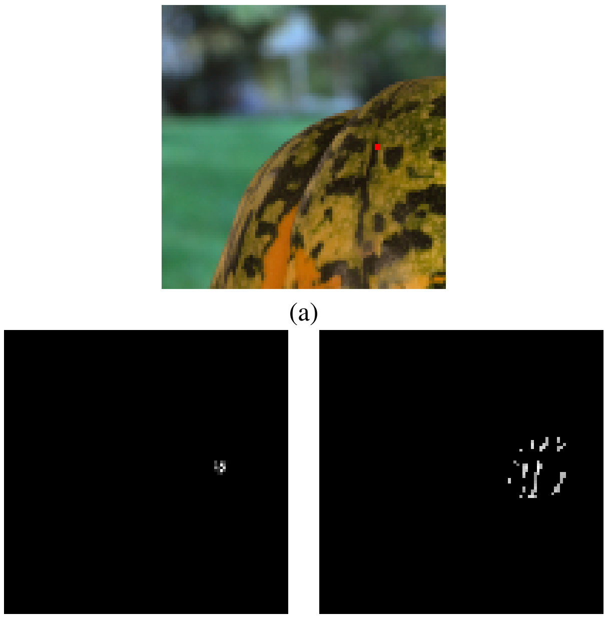
$$D_i\alpha_i \approx k(i,\cdot)^T\boldsymbol{\alpha}, \tag{9}$$

where $\boldsymbol{\alpha}$ is the vector of $\alpha$ values over the image. Really equation (9) replaces the color-line sparsity assumption of [10] as we explain later. It follows that

$$D\boldsymbol{\alpha} \approx A\boldsymbol{\alpha}, \text{ where} \tag{10}$$

$$A = \begin{bmatrix} k(1,1)\cdots k(1,N) \\ \vdots \\ k(N,1)\cdots k(N,N) \end{bmatrix}, \text{ and} \tag{11}$$

$$D = \begin{bmatrix} D_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & D_N \end{bmatrix}. \tag{12}$$


(a)

(b) (c)

Figure 4. (a) Input image, pixel (76,49) highlighted in red. (b) Affinities to pixel (76,49) in [10] (c) Nonlocal affinities to pixel (76,49).

Therefore, $(D-A)\boldsymbol{\alpha} \approx \mathbf{0}$, and $\boldsymbol{\alpha}^T(D-A)^T(D-A)\boldsymbol{\alpha} \approx 0$. What this means is that we can effectively have a matting Laplacian based on the nonlocal principle $L_c = (D - A)^T(D - A)$.

To compare with [10], by expanding and summing (3), Levin *et al.*'s affinity has the form

$$A^L_{ij} = \sum_{\{k|i,j\in S(k)\}} \left(1 + \frac{(I(i) - \mu_k)(I(j) - \mu_k)}{\lambda + \sigma_k^2}\right), \quad (13)$$

where $\mu_k$ and $\sigma_k^2$ are the mean and variance of patch $k$. From this, we can see that the affinity is nonzero only in a spatial region (when $i$ and $j$ are both in the same spatial patch $k$), and there is no measure of texture similarity since (13) simply compares single pixel values $I(i)$ and $I(j)$. See Fig. 4 for visual comparison of (13) and (11).

Due to (11), our affinity includes texture information, has little spatial regularization, and is nonlinear. Further, since we expect sparsity from the nonlocal principle, $L_c$ should exhibit the property of having a small number of clusters for some permutation matrix $P$. For this reason, we refer to this $L_c$ as the nonlocal *clustering* Laplacian.

To extract clusters from $L_c$, we adapt the method described in [13]. First, define

$$M = D^{\frac{-1}{2}} L_c D^{\frac{-1}{2}}. \quad (14)$$

Then, calculate the $m$ smallest eigenvectors of $M$ (we take $m = 10$), $\{V_1, \ldots, V_m\}$. Now, we create a matrix $U$ such that each column corresponds to an eigenvector, but normalized such that each row has unit length under the 2-norm. In this way, each pixel is associated with a type of feature vector with $m$ components. Then, we can treat $U$ as the data matrix in the $k$-means clustering algorithm [7]. So, we use $k$-means to do the final segmentation to locate the clusters. Parameter $k$ controls the final number of clusters and depends on the image complexity.

The key observation is that the number of clusters $k$ that we need to describe the alpha matte is only on the order of 10 to 20, even when the number of pixels is tens of thousands. According to the observation in (4), we only need the user to label $k$ pixels in the entire image. This is in contrast to Levin *et al.*'s method, where their Laplacian does not cluster very well (Fig. 2), especially for small $k$, requiring dense user input in the form of large scribble regions, or even a nearly complete trimap.

There is one technical detail that prevents us from directly using our Laplacian in the computation of the alpha matte, and that is the use of the spatial patches in the nonlocal kernel $k(i, j)$. If we directly use this method, then we lose accuracy near the alpha matte edges on the order of the radius of the spatial patch, which is typically 2-3 pixels. Therefore, we need to construct another Laplacian that can operate on the matte edges to recover fine details. The following section develops such a Laplacian.

## 4. Nonlocal Matting

To maintain accuracy of the matte near the edges, we present a modification to Levin *et al.*'s algorithm. First, some notation. If $C = \{\tau_1, \ldots, \tau_n\}$, define $\boldsymbol{\alpha}(C)$ to be a column vector of length $n$: $\left[\alpha_{\tau_1}, \ldots, \alpha_{\tau_n}\right]^T$. Unless otherwise specified, vectors are column vectors. $S(i)$ represents a spatial neighborhood of pixel $i$ (patch), and $N(i)$ represents the nonlocal neighborhood of pixel $i$.

Suppose

$$\begin{aligned} \mathbf{Q}\boldsymbol{\alpha}(N(i)) &\approx \mathbf{Q}\left[X(N(i)), \mathbf{1}\right] \begin{bmatrix} \boldsymbol{\beta} \\ \beta_0 \end{bmatrix} \quad (15) \\ &\triangleq X_0(N(i)) \begin{bmatrix} \boldsymbol{\beta} \\ \beta_0 \end{bmatrix}, \end{aligned}$$

where $X$ is a $N \times d$ vector of image features with $N$ being the number of pixels, and $d$ being the number of features or channels (we use RGB or grayscale values), and $\mathbf{Q} = \mathrm{diag}(k(i, \cdot)/D_i)$, is a weighting matrix containing the nonlocal data distribution. If we assume $\alpha$ is fixed, the optimal linear parameters are

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\beta}^* \\ \beta_0^* \end{bmatrix} &= \left(X_0^T(N(i))\mathbf{Q}X_0(N(i)) + \lambda \begin{bmatrix} \mathbf{I}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\right)^{-1} \\ &\quad X_0^T(N(i))\mathbf{Q}\boldsymbol{\alpha}(N(i)) \\ &\triangleq X_0^\dagger(i, \mathbf{Q}, \lambda)\boldsymbol{\alpha}(N(i)). \quad (16) \end{aligned}$$

Here, $\lambda$ is a small number ($10^{-5}$ or so) whose purpose is to ensure that the eigenvalues of $X^T(N(i))\mathbf{Q}X(N(i))$ are strictly positive so that the pseudoinverse defined in equation (16) exists and is unique.

Since we now have

$$\begin{aligned}\boldsymbol{\alpha}(N(i)) &\approx X_0(N(i))X_0^{\dagger}(i,\mathbf{Q},\lambda)\boldsymbol{\alpha}(N(i)) \quad (17)\\ &\triangleq \mathbf{B}_i\boldsymbol{\alpha}(N(i)), \quad (18)\end{aligned}$$

we can construct the quadratic form

$$\begin{aligned}\boldsymbol{\alpha}^T L_m \boldsymbol{\alpha} &\triangleq q(\boldsymbol{\alpha}) \quad (19)\\ &= \sum_{i=1}^{N} \boldsymbol{\alpha}(N(i))^T(\mathbf{I}-\mathbf{B}_i)^T(\mathbf{I}-\mathbf{B}_i)\boldsymbol{\alpha}(N(i)). \quad (20)\end{aligned}$$

The difference from [10] is that their data matrices $\mathbf{X}_0$ are local, and weighted by a uniform data matrix $\mathbf{Q}=c\mathbf{I}$, while our data matrices are nonlocal, and appropriately weighted by the nonlocal kernel, biasing the solution toward our assumption (8), and inserting texture information into the algorithm.

To find $\boldsymbol{\alpha}$, we solve the following optimization:

$$\begin{aligned}\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \quad & q(\boldsymbol{\alpha})\\ \text{s.t.} \quad & \boldsymbol{\alpha}(C) = \mathbf{k}\end{aligned}$$

where $C$ is the set of user-constrained pixels and $\mathbf{k}$ is the vector of constrained values. This can be solved exactly and in closed form by a smaller quadratic program as described by Lemma 1 (Sec. 9). Solving this way is faster and more accurate than the brute force method advocated in [21]. Since we use this Laplacian $L_m$ for matte extraction, we refer to it as the nonlocal matting *extraction* Laplacian.

The result of this nonlocal modification is very important. First, by exploiting the nonlocal kernel, we utilize some texture information on the image, effectively grouping pixels with similar texture. Second, the neighborhoods used in calculating the Laplacian are essentially adaptively varying in size as shown in Fig. 3, removing the highly spatial clustering effects of previous methods (see Fig. 2.b), and therefore reducing the amount of input needed. Another advantage is that the nonlocal neighborhoods can safely be much larger. The effect of large neighborhoods is interpreted by [8] as a large point spread function on the matte, causing convergence of iterative solvers to greatly speed up. Conversely, if the pixel's patch is not very similar to any other in the image, the nonlocal neighborhood will be very small and convergence there will be slow. Therefore, an iterative solver can provide feedback to the user based on the local convergence rates about the quality of the final matte, and where he or she should provide additional input.

## 5. Importance of Nonlocality

To give mathematical rigor to the algorithm we have proposed, we provide the following explaination. Suppose that we treat the $\boldsymbol{\alpha}$ matte and image as random variables.

$$X(i) = U(i) + n(i), \quad (21)$$

where U is the underlying true image, and $n$ is white Gaussian noise. Then, we wish to find the best linear approximator (for simplicity, we assume a grayscale image)

$$\hat{\alpha}_i \triangleq \beta X(i) + \beta_0, \quad (22)$$

where $\hat{\alpha}_i$ is the linear estimator of the true $\alpha_i$. By the orthogonality principle, we need

$$E\left[X(i)(\hat{\alpha}_i - \alpha_i)\right] = 0, \quad (23)$$

$$\begin{aligned}E\left[X(i)(\hat{\alpha}_i - \alpha_i)\right] =& \beta E\left[X(i)^2\right] + \beta_0 E\left[X(i)\right]\\ &- E\left[\alpha_i X(i)\right]. \quad (24)\end{aligned}$$

Thanks to the nonlocal principle, we have the discrete approximation

$$P\left[X(i)=X(j)\right] = k(i,j)\frac{1}{D_i} \triangleq W_{(i,j)}. \quad (25)$$

So, we have

$$\begin{aligned}& E\left[X(i)(\hat{\alpha}_i - \alpha_i)\right] = 0\\ & \Leftrightarrow\\ & \beta \sum_{j\in N(i)} W_{(i,j)}X(j)^2 + \beta_0 \sum_{j\in N(i)} W_{(i,j)}X(j)\\ & = E\left[\alpha_i X(i)\right]. \quad (26)\end{aligned}$$

To take the expectation on the right hand side, we need to know the joint distribution $P[\alpha_i, X(i)]$. By Bayes' rule, we have

$$\begin{aligned}& P\left[\alpha_i = k, X(i)=X(j)\right] =\\ & \quad P\left[\alpha_i = k \mid X(i)=X(j)\right] P\left[X(i)=X(j)\right]. \quad (27)\end{aligned}$$

Suppose we assume that $P[\alpha_i = k \mid X(i) = X(j)]$ is such that

$$E\left[\alpha_i \mid X(i)=X(j)\right] = \alpha_j. \quad (28)$$

In other words, pixels sharing the same appearance should be expected to share the same alpha value, which seems reasonable. Then, we have

$$\begin{aligned}& \beta \sum_{j\in N(i)} W_{(i,j)}X(j)^2 + \beta_0 \sum_{j\in N(i)} W_{(i,j)}X(j)\\ & = \sum_{j\in N(i)} W_{(i,j)}\alpha_j X(j). \quad (29)\end{aligned}$$

Since these variables are not zero-mean, the orthogonality principle also requires

$$E\left[\hat{\alpha}_i - \alpha_i\right] = 0. \tag{30}$$

Due to the assumption in equation (28) and the law of iterated expectation, this is equivalent to

$$\beta \sum_{j \in N(i)} W_{(i,j)} X(j) + \beta_0 = \sum_{j \in N(i)} W_{(i,j)} \alpha_j. \tag{31}$$

Equations (29) and (31) *exactly* give us the formulation we arrived at in equation (16). To see why, for grayscale, equation (16) implies we need to satisfy

$$\begin{bmatrix} \sum_{j \in N(i)} W_{(i,j)} X(j)^2 & \sum_{j \in N(i)} W_{(i,j)} X(j) \\ \sum_{j \in N(i)} W_{(i,j)} X(j) & 1 \end{bmatrix} \begin{bmatrix} \beta \\ \beta_0 \end{bmatrix} = \begin{bmatrix} \sum_{j \in N(i)} W_{(i,j)} \alpha_j X(j) \\ \sum_{j \in N(i)} W_{(i,j)} \alpha_j \end{bmatrix}, \tag{32}$$

and the solution to $\begin{bmatrix} \beta \\ \beta_0 \end{bmatrix}$ is the same as in (16).

In summary, we dispose of the color line model requirement in [10] by using the following three new assumptions:

1. The nonlocal principle applies to the alpha matte as in equation (8).
2. The conditional distribution on $\boldsymbol{\alpha}$ given $X$ is such that $E\left[\alpha_i \mid X(i) = X(j)\right] = \alpha_j$.
3. We use the best possible linear predictor for $\boldsymbol{\alpha}$ according to the orthogonality principle.

It is now easy to see that we obtain sparsity and denoising for free. Notice the last column of $X_0^T Q X_0$ always contains the denoised pixel and $X_0^T Q \boldsymbol{\alpha}$ contains the denoised $\alpha$ value.

In this derivation, it is also easy to see why we get better results than Levin *et al.*'s method: they are not using any prior term for the image, while we are using a quite valid and effective image prior. However, notice that the discrete approximation to the image distribution is a sampling. If we have very few similar patches available in the image, then the sampled approximation may be poor, and we may not produce satisfactory results for those pixels. However, we can use this knowledge to tell the user that he or she should manually constrain such pixels (where $D_i$ is small).

## 6. Implementation Details

The proposed algorithm is a hybrid of the nonlocal clustering and matting Laplacians we have defined above. To reduce the user input requirements, we first use the nonlocal clustering Laplacian to extract a small set of clusters as described in Sec. 3. Specifically, the parameters we found to be most satisfactory were $h_1^2 = 10^{-2}$ or sometimes $10^{-3}$ and $h_2^2 = 3$, with $S(i)$ having a radius of 3 pixels. Since it is impractical to compute the kernel function $k(i,j)$ (equation (6)) for all pixels $j$, we restrict the computation to be over a spatial patch of pixel $i$ with radius $3h_2^2$, because the spatial term will force the kernel values close to zero for anything outside the patch. Therefore, we set $k(i,j) = 0$ for $j$ outside the patch. After this, we also only retain the top 30 values for $k(i,\cdot)$ and set the rest to zero. The reason is due to the effect of small, but nonzero, values of $k$ on the sparsity pattern of the nonlocal clustering Laplacian. We observe that keeping more than the top 30 or so kernel values for each pixel does not significantly improve the results, but greatly reduces the sparsity of $L_c$, increasing memory and CPU requirements. Also, rather than computing the kernel patch by patch, we note that it is far more efficient to use locality-sensitive hashing [4] to perform this task. In the nonlocal matting Laplacian, we also perform the same restriction process to the nonlocal neighborhood $N(i)$ for the same reasons.

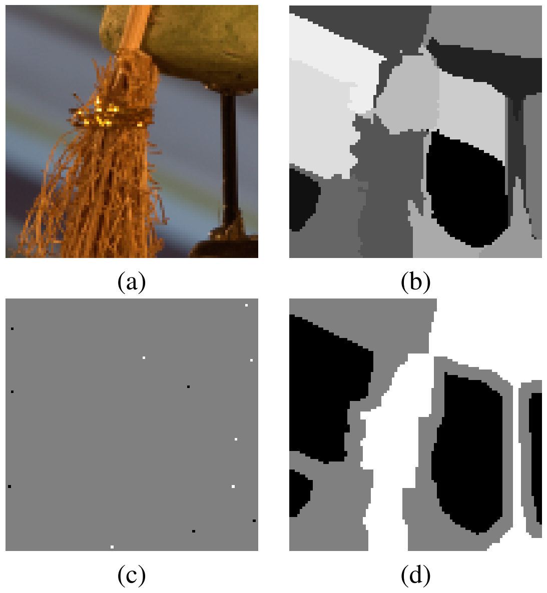


Figure 5. Trimap generation. (a) Input image. (b) Clusters. (c) Sparse user input. (d) Resulting trimap.

For clustering, we typically find that 15 clusters is enough to properly segment the image into the foreground/background regions. However, sometimes as few as 5 will do the job. In cases where the foreground or background is very complex, 30 may be needed. After clustering, we simply ask the user to manually label each cluster that is definitely foreground or background with a single click. The clusters that the user does not label are treated as unknown in the trimap. Since the clustering Laplacian is only accurate to the radius of $S(i)$, we erode any foreground/background edges with a structuring element of the same radius. The result of this process is a trimap as shown in Fig. 5.

Using this trimap, we can then feed it to any appropriate matting method, such as the method based on our own nonlocal matting Laplacian, or [10, 11, 18, 17] etc. We choose our proposed method here to demonstrate its performance.

## 7. Experimental Results and Discussion

To formally evaluate the algorithm we have developed, we test its accuracy on the dataset given by [16] by the mean squared error (MSE) as well as by visual inspection of matte quality. We use the algorithm in [10] for comparison, since it is the state of the art with respect to sparse user input. We provide it with the same user input as to our algorithm.

The results for the MSE are shown in Fig. 6, where we can immediately see a drastic reduction in MSE. In fact, the average ratio of the Levin *et al.*'s MSE to the nonlocal MSE is 4.04 with a sample standard deviation of 1.0. Therefore, we can say that our method outperforms Levin *et al.*'s with confidence $> 99.99\%$ when the input is sparse. Perhaps even more important is the observation that our MSE curve is much smoother, suggesting our method is more robust to variations in user input.

However, the MSE does not have a linear correlation to human perception. So, we need to visually verify that the extracted mattes are high quality despite the sparse input. For this, we provide several examples in Fig. 7 (more at our website). First, it is obvious that the amount of input required for our method is miniscule. The effects of spatial regularization can be seen as oversmoothing in column (c), while our results in (d) remain crisp. In column (e), background replacement highlights the limitations of [10], showing significant bleed-through, while ours in column (f) appears much cleaner. It is clear that even with such sparse input, the matte quality is much higher with our nonlocal matting.

## 8. Conclusion

The algorithm we describe in this paper represents a significant leap in matting literature. We have demonstrated that the nonlocal principle can be extended to the alpha matte in a way to dramatically reduce human effort, while also preserving accuracy. Furthermore, we have answered the question of what is good user input and how to best guide the user. The obvious application is the creation of an interactive matting system that is much quicker and easier than any existing method. More interestingly, since we show that creating an accurate matte depends only on labelling on the order of 10 pixels, this work is more amenable than the existing methods to automatic and semi-automatic matting.

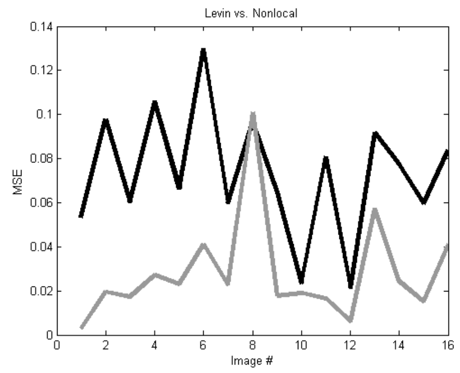


Figure 6. MSE of Levin *et al.*'s method vs. our method with the same amount of input on images from [16]. [10] = black, nonlocal = gray.

## Acknowledgement

This work was supported in part by National Science Foundation grant IIS-0347877, IIS-0916607, and US Army Research Laboratory and the US Army Research Office under grant ARO W911NF-08-1-0504.

## 9. Appendix

**Lemma 1.** *Suppose* $f(x) = \frac{1}{2}x^T Hx + c^T x$, *for some* $x, c \in \mathbb{R}^n$, *and* $H$ *positive semi-definite. Further suppose* $P$ *is a permutation matrix such that* $y = Px = [y_A, y_B]^T$, *where* $y_A = [x_{I_1}, \ldots, x_{I_m}]^T$, *and* $b = Pc$, $G = PHP^{-1} = \begin{bmatrix} G_1, G_2 \\ G_3, G_4 \end{bmatrix}$, *where* $G_1$ *is* $m \times m$. *Then, the optimal solution to*

$$\min_x \quad f(x) \quad \text{s.t.} \quad x_{I_j} = k_j \quad \forall j \in [1, m]$$

*is* $x^* = P^{-1}y^*$, *where* $y_A^* = [k_1, k_2, \ldots]^T$, *and* $y_B^* = -G_4^{-1}(G_3 y_A + b_B)$.

*Proof.* Observe $f(x) = \frac{1}{2}y^T Gy + b^T y$.

$$f(x) = \frac{1}{2}\begin{bmatrix} y_A^T, y_B^T \end{bmatrix} \begin{bmatrix} G_1, G_2 \\ G_3, G_4 \end{bmatrix} \begin{bmatrix} y_A \\ y_B \end{bmatrix} + \begin{bmatrix} b_A^T, b_B^T \end{bmatrix} \begin{bmatrix} y_A \\ y_B \end{bmatrix} \tag{33}$$

$$= \frac{1}{2}y_B^T G_4 y_B + \left(y_A^T G_2 + b_B^T\right) y_B + \frac{1}{2}y_A^T G_1 y_A + b_A^T y_A. \tag{34}$$

Note this is a quadratic program in $y_B$, and since

$$\frac{\partial}{\partial y_B} f = G_4 y_B + (G_3 y_A + b_B), \tag{35}$$

$$y_B^* = -G_4^{-1}(G_3 y_A + b_B). \quad \square \tag{36}$$

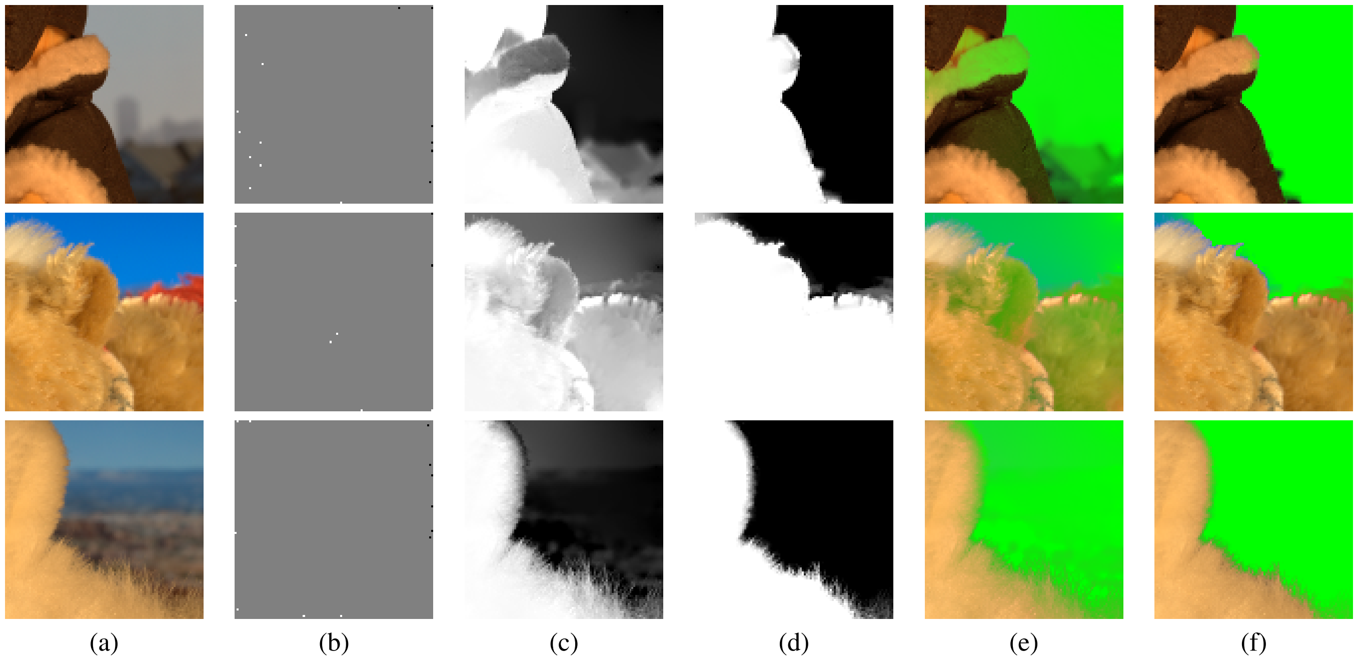


Figure 7. Visual inspection of matte quality. (a) Input image. (b) Sparse user input. (c) Levin *et al.*'s algorithm [10]. (d) Nonlocal algorithm. (e) Background replacement with (c). (f) Background replacement with (d). Input is placed near the image border to exaggerate the problems caused by spatial regularization.

## References

[1] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007. 2193

[2] A. Buades, B. Coll, and J. Morel. A non-local algorithm for image denoising. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, volume 2, 2005. 2194, 2195

[3] Y. Chuang, B. Curless, D. Salesin, and R. Szeliski. A bayesian approach to digital matting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2. IEEE Computer Society; 1999, 2001. 2193

[4] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004. 2198

[5] O. Duchenne, J. Audibert, R. Keriven, J. Ponce, and F. Segonne. Segmentation by transduction. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, pages 1–8, 2008. 2193, 2194

[6] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann. Random walks for interactive alpha-matting. In *Proceedings of VIIP*, volume 2005, pages 423–429. Citeseer, 2005. 2193

[7] J. Hartigan and M. Wong. A k-means clustering algorithm. *JR Stat. Soc., Ser. C*, 28:100–108, 1979. 2196

[8] K. He, J. Sun, and X. Tang. Fast matting using large kernel matting Laplacian matrices. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2165–2172. IEEE, 2010. 2197

[9] N. Joshi, W. Matusik, and S. Avidan. Natural video matting using camera arrays. In *ACM SIGGRAPH 2006 Papers*, pages 779–786. ACM, 2006. 2193

[10] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):228–242, 2008. 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200

[11] A. Levin, A. Rav-Acha, and D. Lischinski. Spectral matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1699–1712, 2008. 2193, 2195, 2199

[12] M. McGuire, W. Matusik, H. Pfister, J. Hughes, and F. Durand. Defocus video matting. *ACM Transactions on Graphics (TOG)*, 24(3):567–576, 2005. 2193

[13] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14: Proceeding of the 2001 Conference*, pages 849–856, 2001. 2196

[14] E. Reinhard and E. Khan. Depth-of-field-based alpha-matte extraction. In *Proceedings of the 2nd symposium on Applied perception in graphics and visualization*, pages 95–102. ACM, 2005. 2193

[15] C. Rhemann, C. Rother, A. Rav-Acha, and T. Sharp. High resolution matting via interactive trimap segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, pages 1–8, 2008. 2193

[16] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott. A perceptually motivated online benchmark for image matting. In *Proc. CVPR*, pages 1826–1833. Citeseer, 2009. 2199

[17] D. Singaraju, C. Rother, and C. Rhemann. New appearance models for natural image matting. 2009. 2199

[18] J. Sun, J. Jia, C. Tang, and H. Shum. Poisson matting. In *International Conference on Computer Graphics and Interactive Techniques*, pages 315–321. ACM New York, NY, USA, 2004. 2193, 2199

[19] J. Wang and M. Cohen. An iterative optimization approach for unified image segmentation and matting. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 936–943. IEEE, 2005. 2193

[20] O. Wang, J. Finger, Q. Yang, J. Davis, and R. Yang. Automatic natural video matting with depth. In *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on*, pages 469–472. IEEE, 2007. 2193

[21] Y. Zheng and C. Kambhamettu. Learning Based Digital Matting. In *ICCV*, 2009. 2197