

# A Closed Form Solution to Natural Image Matting

Anat Levin Dani Lischinski Yair Weiss

School of Computer Science and Engineering

The Hebrew University of Jerusalem

*{alevin,danix,yweiss}@cs.huji.ac.il}*

## Abstract

*Interactive digital matting, the process of extracting a foreground object from an image based on limited user input, is an important task in image and video editing. From a computer vision perspective, this task is extremely challenging because it is massively ill-posed — at each pixel we must estimate the foreground and the background colors, as well as the foreground opacity (“alpha matte”) from a single color measurement. Current approaches either restrict the estimation to a small part of the image, estimating foreground and background colors based on nearby pixels where they are known, or perform iterative nonlinear estimation by alternating foreground and background color estimation with alpha estimation.*

*In this paper we present a closed form solution to natural image matting. We derive a cost function from local smoothness assumptions on foreground and background colors, and show that in the resulting expression it is possible to analytically eliminate the foreground and background colors to obtain a quadratic cost function in alpha. This allows us to find the globally optimal alpha matte by solving a sparse linear system of equations. Furthermore, the closed form formula allows us to predict the properties of the solution by analyzing the eigenvectors of a sparse matrix, closely related to matrices used in spectral image segmentation algorithms. We show that high quality mattes can be obtained on natural images from a surprisingly small amount of user input.*

## 1. Introduction

Natural image matting and compositing is of central importance in image and video editing. Formally, image matting methods take as input an image  $I$ , which is assumed to be a composite of a foreground image  $F$  and a background image  $B$ . The color of the  $i$ -th pixel is assumed to be a linear combination of the corresponding foreground and background colors,

$$I_i = \alpha_i F_i + (1 - \alpha_i) B_i, \quad (1)$$

where  $\alpha_i$  is the pixel’s foreground opacity. In natural image matting, all quantities on the right hand side of the *compositing equation* (1) are unknown. Thus, for a 3 channel

color image, at each pixel there are 3 equations and 7 unknowns.

Obviously, this is a severely under-constrained problem, and user interaction is required to extract a good matte. Most recent methods expect the user to provide a *trimap* as a starting point; an example is shown in Figure 1(e). The trimap is a rough (typically hand-drawn) segmentation of the image into three regions: foreground (shown in white), background (shown in black) and unknown (shown in gray). Given the trimap, these methods typically solve for  $F$ ,  $B$ , and  $\alpha$  simultaneously. This is typically done by iterative nonlinear optimization, alternating the estimation of  $F$  and  $B$  with that of  $\alpha$ . In practice, this means that for good results the unknown regions in the trimap must be as small as possible. As a consequence, trimap-based approaches typically experience difficulty handling images with a significant portion of mixed pixels or when the foreground object has many holes [15]. In such challenging cases a great deal of experience and user interaction may be necessary to construct a trimap that would yield a good matte. Another problem with the trimap interface is that the user cannot directly influence the matte in the most important part of the image: the mixed pixels.

In this paper we present a new closed-form solution for extracting the alpha matte from a natural image. We derive a cost function from local smoothness assumptions on foreground and background colors  $F$  and  $B$ , and show that in the resulting expression it is possible to analytically eliminate  $F$  and  $B$ , yielding a quadratic cost function in  $\alpha$ . The alpha matte produced by our method is the global optimum of this cost function, which may be obtained by solving a sparse linear system. Since our approach computes  $\alpha$  directly and without requiring reliable estimates for  $F$  and  $B$ , a surprisingly small amount of user input (such as a sparse set of scribbles) is often sufficient for extracting a high quality matte. Furthermore, our closed-form formulation enables one to understand and predict the properties of the solution by examining the eigenvectors of a sparse matrix, closely related to matrices used in spectral image segmentation algorithms. In addition to providing a solid theoretical basis for our approach, such analysis can provide useful hints to the user regarding where in the image scribbles should be placed.

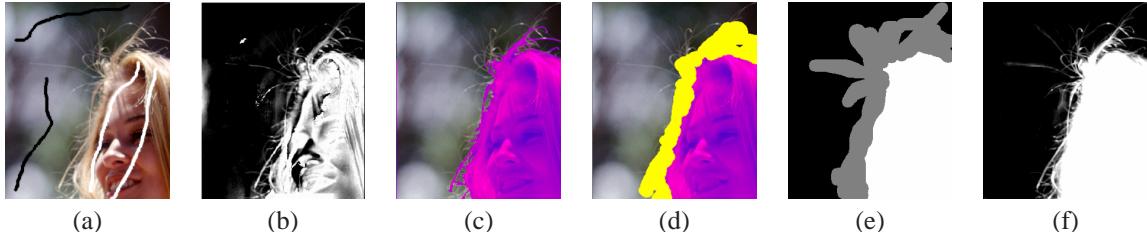


Figure 1. (a) An image with sparse constraints: white scribbles indicate foreground, black scribbles indicate background. Applying Bayesian matting to such sparse input produces a completely erroneous matte (b). Foreground extraction algorithms, such as [9, 11] produce a hard segmentation (c). An automatically generated trimap from a hard segmentation may miss fine features (d). An accurate hand-drawn trimap (e) is required in this case to produce a reasonable matte (f). (Images taken from [15])

## 1.1. Previous work

Most existing methods for natural image matting require the input image to be accompanied by a *trimap* [1, 2, 4, 5, 12, 14], labeling each pixel as foreground, background, or unknown. The goal of the method is to solve the compositing equation (1) for the unknown pixels. This is typically done by exploiting some local regularity assumptions on  $F$  and  $B$  to predict their values for each pixel in the unknown region. In the Corel KnockOut algorithm [2],  $F$  and  $B$  are assumed to be smooth and the prediction is based on a weighted average of known foreground and background pixels (closer pixels receive higher weight). Some algorithms [5, 12] assume that the local foreground and background come from a relatively simple color distribution. Perhaps the most successful of these algorithms is the Bayesian matting algorithm [5], where a mixture of oriented Gaussians is used to learn the local distribution and then  $\alpha$ ,  $F$ , and  $B$  are estimated as the most probable ones given that distribution. Such methods work well when the color distributions of the foreground and the background do not overlap, and the unknown region in the trimap is small. As demonstrated in Figure 1(b) a sparse set of constraints could lead to a completely erroneous matte. In contrast, while our approach also makes certain smoothness assumptions regarding  $F$  and  $B$ , it does not involve estimating the values of these functions until after the matte has been extracted.

The Poisson matting method [14], also expects a trimap as part of its input, and computes the alpha matte in the mixed region by solving a Poisson equation with the matte gradient field and Dirichlet boundary conditions. In the *global Poisson matting* method the matte gradient field is approximated as  $\nabla I/(F - B)$  by taking the gradient of the compositing equation, and neglecting the gradients in  $F$  and  $B$ . The matte is then found by solving for a function whose gradients are as close as possible to the approximated matte gradient field. Whenever  $F$  or  $B$  are not sufficiently smooth inside the unknown region, the resulting matte might not be correct, and additional local manipulations may need to be applied interactively to the matte gradient field in order to obtain a satisfactory solution. This interactive refinement process is referred to as *local Poisson matting*. As we shall

see, our method makes weaker assumptions on the behavior of  $F$  and  $B$ , which generally leads to more accurate mattes.

Recently, several successful approaches for extracting a foreground object from its background have been proposed [3, 9, 11]. Both approaches translate simple user-specified constraints (such as scribbles, or a bounding rectangle) into a min-cut problem. Solving the min-cut problem yields a hard binary segmentation, rather than a fractional alpha matte (Figure 1(c)). The hard segmentation could be transformed into a trimap by erosion, but this could still miss some fine or fuzzy features (Figure 1(d)). Although Rother *et al.* [11] do perform border matting by fitting a parametric alpha profile in a narrow strip around the hard boundary, this is more akin to feathering than to full alpha matting, since wide fuzzy regions cannot be handled in this manner.

Our approach is closely related to the colorization method of Levin *et al.* [7], and the random walk alpha matting method of Grady *et al.* [6]. Both of these methods propagate scribbled constraints to the entire image by minimizing a quadratic cost function. Here we apply a similar strategy, but our assumptions and cost function are modified so as to better suit the matting problem.

Another scribble-based interface for interactive matting was recently proposed by Wang and Cohen [15]. Starting from a few scribbles indicating a small number of background and foreground pixels, they use belief propagation to iteratively estimate the unknowns at every pixel in the image. While this approach has produced some impressive results, it has the disadvantage of employing an expensive iterative non-linear optimization process, which might converge to different local minima.

## 2. Derivation

For clarity of exposition we begin by deriving a closed-form solution for alpha matting of grayscale images. This solution will then be extended to the case of color images in Section 2.1.

As mentioned earlier, the matting problem is severely under-constrained. Therefore, some assumptions on the nature of  $F$ ,  $B$  and/or  $\alpha$  are needed. To derive our solution for the grayscale case we make the assumption that both  $F$  and  $B$  are approximately constant over a small window around

each pixel. Note that assuming  $F$  and  $B$  are locally smooth does not mean that the input image  $I$  is locally smooth, since discontinuities in  $\alpha$  can account for the discontinuities in  $I$ . This assumption, which will be somewhat relaxed in Section 2.1, allows us to rewrite (1) expressing  $\alpha$  as a linear function of the image  $I$ :

$$\alpha_i \approx aI_i + b, \quad \forall i \in w, \quad (2)$$

where  $a = \frac{1}{F-B}$ ,  $b = -\frac{B}{F-B}$  and  $w$  is a small image window. This linear relation is similar to the prior used in [17]. Our goal in this paper will be to find  $\alpha$ ,  $a$  and  $b$  minimizing the cost function

$$J(\alpha, a, b) = \sum_{j \in I} \left( \sum_{i \in w_j} (\alpha_i - a_j I_i - b_j)^2 + \epsilon a_j^2 \right), \quad (3)$$

where  $w_j$  is a small window around pixel  $j$ .

The cost function above includes a regularization term on  $a$ . One reason for adding this term is numerical stability. For example, if the image is constant in the  $j$ -th window,  $a_j$  and  $b_j$  cannot be uniquely determined without a prior. Also, minimizing the norm of  $a$  biases the solution towards smoother mattes (since  $a_j = 0$  means that  $\alpha$  is constant over the  $j$ -th window).

In our implementation, we typically use windows of  $3 \times 3$  pixels. Since we place a window around each pixel, the windows  $w_j$  in (3) overlap. It is this property that enables the propagation of information between neighboring pixels. The cost function is quadratic in  $\alpha$ ,  $a$  and  $b$ , with  $3N$  unknowns for an image with  $N$  pixels. Fortunately, as we show below,  $a$  and  $b$  may be eliminated from (3), leaving us with a *quadratic* cost in only  $N$  unknowns: the alpha values of the pixels.

**Theorem 1** Define  $J(\alpha)$  as

$$J(\alpha) = \min_{a, b} J(\alpha, a, b).$$

Then

$$J(\alpha) = \alpha^T L \alpha, \quad (4)$$

where  $L$  is an  $N \times N$  matrix, whose  $(i, j)$ -th entry is:

$$\sum_{k|(i,j) \in w_k} \left( \delta_{ij} - \frac{1}{|w_k|} \left( 1 + \frac{1}{\frac{\epsilon}{|w_k|} + \sigma_k^2} (I_i - \mu_k)(I_j - \mu_k) \right) \right) \quad (5)$$

Here  $\delta_{ij}$  is the Kronecker delta,  $\mu_k$  and  $\sigma_k^2$  are the mean and variance of the intensities in the window  $w_k$  around  $k$ , and  $|w_k|$  is the number of pixels in this window.

*Proof:* Rewriting (3) using matrix notation we obtain

$$J(\alpha, a, b) = \sum_k \left\| G_k \begin{bmatrix} a_k \\ b_k \end{bmatrix} - \bar{\alpha}_k \right\|^2, \quad (6)$$

where for every window  $w_k$ ,  $G_k$  is defined as a  $(|w_k|+1) \times 2$  matrix. For each  $i \in w_k$ ,  $G_k$  contains a row of the form  $[I_i, 1]$ , and the last row of  $G_k$  is of the form  $[\sqrt{\epsilon}, 0]$ . For a given matte  $\alpha$  we define  $\bar{\alpha}_k$  as a  $(|w_k|+1) \times 1$  vector, whose entries are  $\alpha_i$  for every  $i \in w_k$ , and whose last entry is 0. The elements in  $\bar{\alpha}_k$  and  $G_k$  are ordered correspondingly.

For a given matte  $\alpha$  the optimal pair  $a_k^*, b_k^*$  inside each window  $w_k$  is the solution to the least squares problem:

$$(a_k^*, b_k^*) = \operatorname{argmin} \left\| G_k \begin{bmatrix} a_k \\ b_k \end{bmatrix} - \bar{\alpha}_k \right\| \quad (7)$$

$$= (G_k^T G_k)^{-1} G_k^T \bar{\alpha}_k \quad (8)$$

Substituting this solution into (6) and denoting  $\tilde{G}_k = I - G_k(G_k^T G_k)^{-1} G_k^T$  we obtain

$$J(\alpha) = \sum_k \bar{\alpha}_k^T \tilde{G}_k^T \tilde{G}_k \bar{\alpha}_k,$$

and some further algebraic manipulations show that the  $(i, j)$ -th element of  $\tilde{G}_k^T \tilde{G}_k$  may be expressed as:

$$\delta_{ij} - \frac{1}{|w_k|} \left( 1 + \frac{1}{\frac{\epsilon}{|w_k|} + \sigma_k^2} (I_i - \mu_k)(I_j - \mu_k) \right).$$

Summing over  $k$  yields the expression in (5).  $\square$

## 2.1. Color Images

A simple way to apply the cost function to color images is to apply the gray level cost to each channel separately. Alternatively we can replace the linear model (2), with a 4D linear model:

$$\alpha_i \approx \sum_c a_i^c I_i^c + b, \quad \forall i \in w \quad (9)$$

The advantage of this combined linear model is that it relaxes our previous assumption that  $F$  and  $B$  are constant over each window. Instead, as we show below, it is enough to assume that in a small window each of  $F$  and  $B$  is a linear mixture of two colors; in other words, the values  $F_i$  in a small window lie on a single line in the RGB color space:  $F_i = \beta_i F_1 + (1 - \beta_i) F_2$ , and the same is true for the background values  $B_i$ . In what follows we refer to this assumption as the *color line model*.

Such a model is useful since it captures, for example, the varying shading on a surface with a constant albedo. Another example is a situation where the window contains an edge between two uniformly colored regions both belonging to the background or the foreground. Furthermore, Omer and Werman [10] demonstrated that in many natural images the pixel colors in RGB space tend to form a relatively small number of elongated clusters. Although these clusters are not straight lines, their skeletons are roughly linear locally.

**Theorem 2** If the foreground and background colors in a window satisfy the color line model we can express

$$\alpha_i = \sum_c a^c I_i^c + b, \quad \forall i \in w.$$

*Proof:* Substituting into (1) the linear combinations  $F_i = \beta_i^F F_1 + (1 - \beta_i^F) F_2$  and  $B_i = \beta_i^B B_1 + (1 - \beta_i^B) B_2$ , where  $F_1, F_2, B_1, B_2$  are constant over a small window, we obtain:

$$I_i^c = \alpha_i (\beta_i^F F_1^c + (1 - \beta_i^F) F_2^c) + (1 - \alpha_i) (\beta_i^B B_1^c + (1 - \beta_i^B) B_2^c).$$

Let  $H$  be a  $3 \times 3$  matrix whose  $c$ -th row is  $[F_2^c + B_2^c, F_1^c - F_2^c, B_1^c - B_2^c]$ . Then the above may be rewritten as:

$$H \begin{bmatrix} \alpha_i \\ \alpha_i \beta_i^F \\ (1 - \alpha_i) \beta_i^B \end{bmatrix} = I_i - B_2,$$

where  $I_i$  and  $B_2$  are  $3 \times 1$  vectors representing 3 color channels. We denote by  $a^1, a^2, a^3$  the elements in the first row of  $H^{-1}$ , and by  $b$  the scalar product of first row of  $H^{-1}$  with the vector  $B_2$ . We then obtain  $\alpha_i = \sum_c a^c I_i^c + b$ .  $\square$

Using the 4D linear model (9) we define the following cost function for matting of RGB images:

$$J(\alpha, a, b) = \sum_{j \in I} \left( \sum_{i \in w_j} \left( \alpha_i - \sum_c a_j^c I_i^c - b_j \right)^2 + \epsilon \sum_c a_j^{c^2} \right) \quad (10)$$

Similarly to the grayscale case,  $a^c$  and  $b$  can be eliminated from the cost function, yielding a quadratic cost in the  $\alpha$  unknowns alone:

$$J(\alpha) = \alpha^T L \alpha. \quad (11)$$

Here  $L$  is an  $N \times N$  matrix, whose  $(i, j)$ -th element is:

$$\sum_{k|(i,j) \in w_k} \left( \delta_{ij} - \frac{1}{|w_k|} \left( 1 + (I_i - \mu_k)(\Sigma_k + \frac{\epsilon}{|w_k|} I_3)^{-1} (I_j - \mu_k) \right) \right) \quad (12)$$

where  $\Sigma_k$  is a  $3 \times 3$  covariance matrix,  $\mu_k$  is a  $3 \times 1$  mean vector of the colors in a window  $w_k$ , and  $I_3$  is the  $3 \times 3$  identity matrix.

We refer to the matrix  $L$  in equations (5) and (12) as the *matting Laplacian*. Note that the elements in each row of  $L$  sum to zero, and therefore the nullspace of  $L$  includes the constant vector. If  $\epsilon = 0$  is used, the nullspace of  $L$  also includes every color channel of  $I$ .

### 3. Constraints and User Interface

In our system the user-supplied constraints on the matte are provided via a scribble-based GUI. The user uses a background brush (black scribbles in our examples) to indicate background pixels ( $\alpha = 0$ ) and a foreground brush (white scribbles) to indicate foreground pixels ( $\alpha = 1$ ).

To extract an alpha matte matching the user's scribbles we solve for

$$\alpha = \operatorname{argmin} \alpha^T L \alpha, \quad \text{s.t. } \alpha_i = s_i, \quad \forall i \in S \quad (13)$$



Figure 2. Matting examples. (a,c) Input images with scribbles. (b,d) Extracted mattes.

where  $S$  is the group of scribbled pixels and  $s_i$  is the value indicated by the scribble.

**Theorem 3** Let  $I$  be an image formed from  $F$  and  $B$  according to (1), and let  $\alpha^*$  denote the true alpha matte. If  $F$  and  $B$  satisfy the color line model in every local window  $w_k$ , and if the user-specified constraints  $S$  are consistent with  $\alpha^*$ , then  $\alpha^*$  is an optimal solution for the system (13), where  $L$  is constructed with  $\epsilon = 0$ .

*Proof:* Since  $\epsilon = 0$ , if the color line model is satisfied in every window  $w_k$ , it follows from the definition (10) that  $J(\alpha^*, a, b) = 0$ , and therefore  $J(\alpha^*) = \alpha^{*T} L \alpha^* = 0$ .  $\square$

We demonstrate this in figure 2. The first image (figure 2(a)) is a synthetic example that was created by compositing computer-simulated (monochromatic) smoke over a simple background with several color bands, which satisfies the color line model. The black and white scribbles show the input constraints. The matte extracted by our method (figure 2(b)) is indeed *identical* to the ground truth matte. The second example (figure 2(c)) is a real image, with fairly uniform foreground and background colors. By scribbling only two black and white points, a high quality matte was extracted (figure 2(d)).

### 4. Spectral Analysis

The matting Laplacian matrix  $L$  is a symmetric positive definite matrix, as evident from theorem 1 and its proof. This matrix may also be written as  $L = D - W$ , where  $D$  is a diagonal matrix  $D(i, i) = \sum_j W(i, j)$  and  $W$  is a symmetric matrix, whose off-diagonal entries are defined by (12). Thus, the matrix  $L$  is the graph Laplacian used in spectral methods for segmentation, but with a novel affinity function given by (12). For comparison, the typical way to define the affinity function (e.g., in normalized cuts image segmentation algorithms [13]) is to set

$$W_G(i, j) = e^{-\|I_i - I_j\|^2 / \sigma^2}, \quad (14)$$

where  $\sigma$  is a global constant (typically chosen by hand). This affinity is large for nearby pixels with similar colors and approaches zero when the color difference is much greater than  $\sigma$ . The random walk matting algorithm [6] has used a similar affinity function for the matting problem, but the color distance between two pixels was taken after applying a linear transformation to their colors. The transformation is image-dependent and is estimated using a manifold learning technique.

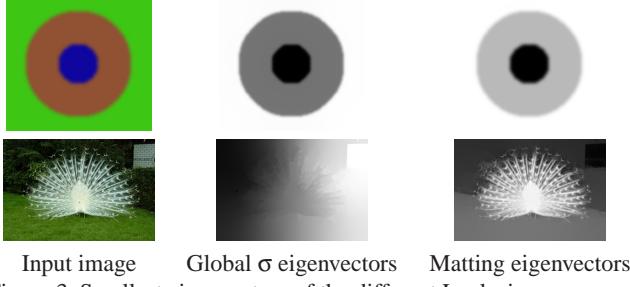


Figure 3. Smallest eigenvectors of the different Laplacians.

In contrast, by rewriting the matting Laplacian as  $L = D - W$  we obtain the following affinity function, which we refer to as “the matting affinity”:

$$W_M(i, j) = \sum_{k|(i,j) \in w_k} \frac{1}{|w_k|} \left( 1 + (I_i - \mu_k)(\Sigma_k + \frac{\epsilon}{|w_k|} I_3)^{-1} (I_j - \mu_k) \right) \quad (15)$$

To gain intuition regarding the matting affinity, consider an image patch containing exactly two colors (e.g., an ideal edge). In this case, it can be shown (see [8] for a proof) that the affinity between two pixels of the *same* color decreases with distance, while the affinity between pixels of *different* colors is zero. In general, we obtain a similar situation to that of standard affinity functions: nearby pixels with similar colors have high affinity, while nearby pixels with different colors have small affinity. However, note that the matting affinity *does not have a global scaling parameter*  $\sigma$  and rather uses local estimates of means and variances. As we show subsequently, this adaptive property leads to a significant improvement in performance. A similar observation was also made in [16], which suggests that local adaptation of the scaling parameter improves image segmentation results.

To compare the two affinity functions we examine the smallest eigenvectors of the corresponding Laplacians, since these eigenvectors are used by spectral segmentation algorithms for partitioning images.

Figure 3 shows the second smallest eigenvector (the first smallest eigenvector is the constant image in both cases) for both Laplacian matrices on two example images. The first example is a simple image with concentric circles of different color. In this case the boundaries between regions are very simple, and both Laplacians capture the transitions correctly. The second example is an image of a peacock. The global  $\sigma$  eigenvector (used by the normalized-cuts algorithm) fails to capture the complex fuzzy boundaries between the peacock’s tail feathers and the background. In contrast, the matting Laplacian’s eigenvector separates the peacock from the background very well. The matting Laplacian in this case was computed with  $\epsilon = 0.0001$ .

#### 4.1. The eigenvectors as guides

While the matting problem is ill-posed without some user input, the matting Laplacian matrix contains a lot of

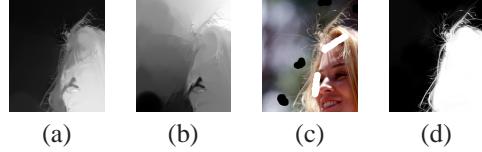


Figure 4. Smallest eigenvectors (a-b) are used for guiding scribble placement (c). The resulting matte is shown in (d).

information on the image even before any scribbles have been provided, as demonstrated in the previous section.

This suggests that looking at the smallest eigenvectors of the matting Laplacian can guide the user *where to place scribbles*. For example, the extracted matte tends to be piecewise constant in the same regions where the smallest eigenvectors are piecewise constant. If the values inside a segment in the eigenvector image are coherent, a single scribble within such a segment should suffice to propagate the desired value to the entire segment. On the other hand, areas where the eigenvector’s values are less coherent correspond to more “difficult” regions in the image, suggesting that more scribbling efforts might be required there.

Stated somewhat more precisely, the alpha matte may be predicted by examining the smaller eigenvectors of the matting Laplacian, since an optimal solution to (13) will be to a large degree spanned by the smaller eigenvectors. In fact (see [8]), it is possible to bound the weight that the optimal solution will assign to larger eigenvectors, as a function of the ratios of the corresponding eigenvalues.

Figure 4 illustrates how a scribbling process may be guided by the eigenvectors. By examining the two smallest eigenvectors (fig 4(a-b)) we placed a scribble inside each region exhibiting coherent eigenvector values (fig 4(c)). The resulting matte is shown in fig 4(d). Note that the scribbles in fig 4(c) were our first, and single attempt to place scribbles on this image.

## 5. Results

We show here results of our closed form solution for extracting alpha mattes by minimizing the matting Laplacian subject to the scribbled constraints. Since the matting Laplacian is quadratic in alpha, the minimum can be found exactly by solving a sparse set of linear equations. We usually define the matting Laplacian using  $3 \times 3$  windows. When the foreground and background color distributions are not very complex using wider windows is helpful. However, using wider windows increases computation time since the resulting system is less sparse. To overcome this, we consider the linear coefficients (eq. 9) that relate an alpha matte to an image. The coefficients obtained using wide windows on a fine resolution image are similar to those obtained with smaller windows on a coarser image. Therefore we can solve for the alpha matte using  $3 \times 3$  windows on a coarse image and compute the linear coefficients which relate it to the coarse image. We then interpolate the lin-

ear coefficients and apply them on a finer resolution image. The alpha matte obtained using this approach is similar to the one that would have obtained by solving the matting system directly on the fine image with wider windows. More details are provided in [8].

For the results shown here we solve the matting system using Matlab’s direct solver (the “backslash” operator) which takes 20 seconds for a 200 by 300 image on a 2.8GHz CPU. Processing big images using the Matlab solver is impossible due to memory limitations. To overcome this we use a coarse-to-fine scheme. We downsample the image and the scribbles and solve in a lower resolution. The reconstructed alpha is then interpolated to the finer resolution, alpha values are thresholded and pixels with alpha close to 0 or 1 are considered constraints in the finer resolution. Constraint pixels can be eliminated from the system, reducing the system size. We have also implemented a multigrid solver for matte extraction. The multigrid solver runs in a couple of seconds even for very large images, but with a small degradation in matte quality.

We show here only the extracted alpha mattes. Note that for compositing on a novel background, we also need to solve for  $F$ . After the matte has been found, it is possible to solve for the  $a^c$  and  $b$  coefficients directly from equation (10) and extract the foreground and background from them. However, we have found that better estimations of foreground and background are obtained by solving a new set of linear equations in  $F$  and  $B$ , derived by introducing some explicit smoothness priors on  $F$  and  $B$  into equation (1). More information on the foreground reconstruction as well as some compositing results can be found in [8].

Figure 5 shows the mattes extracted using our technique on two challenging images used in [15] and compares our results to several other recent algorithms. It can be seen that our results on these examples are comparable in quality to those of [15], even though we use a far simpler algorithm. Global Poisson matting cannot extract a good matte from sparse “scribbles” although its performance with a trimap is quite good. The random walk matting algorithm [6] also minimizes a Laplacian but uses an affinity function with a global scaling parameter and hence has a particular difficulty with the peacock image.

To obtain a more quantitative comparison between the algorithms, we performed an experiment with synthetic composites for which we have the ground truth alpha matte. We randomly extracted 2000 subimages from the image shown in figure 6(h). We used each subimage as a background and composited over it a uniform foreground image using two different alpha mattes: the first matte is a computer simulated smoke most of which is partially transparent; the other matte is a part of a circle, mostly opaque with a feathered boundary. The mattes are shown in figure 6(c). Consequently, we obtained 4000 composite images, two of which are shown in figure 6(a). On this set of images we compared the performance of four matting algorithms — Wang

and Cohen, global Poisson matting, random walk matting, and our own (using  $3 \times 3$  windows with no pyramid). All algorithms were provided a trimap as input. Examples of the trimaps and the results produced by the different methods are shown in figure 6(a,d–g)). For each algorithm, we measured the summed absolute error between the extracted matte and the ground truth. Figures 6(i,j) plot the average error of the four algorithms as a function of the smoothness of the background (specifically we measured the average gradient strength, binned into 10 bins). The errors in the smoke matte are plotted in figure 6(i), while errors in the circle matte are plotted in figure 6(j). When the background is smooth, all algorithms perform well with both mattes. When the background contains strong gradients, global Poisson matting performs poorly (recall that it assumes that background and foreground gradients are negligible). Of the remaining algorithms, our algorithm consistently produced the most accurate results.

Figure 7 shows an example (from [15]), where Wang and Cohen’s method fails to extract a good matte from scribbles due to color ambiguity between the foreground and the background. The same method, however, is able to produce an acceptable matte when supplied with a trimap. Our method produces a cleaner, but also not perfect matte from the same set of scribbles, but adding a small number of additional scribbles results in a better matte.

Figure 8 shows another example (a closeup of the Koala image from [14]), where there’s an ambiguity between foreground and background colors. In this case the matte produced by our method is clearly better than the one produced by the Wang-Cohen method. To better understand why this is the case, we show an RGB histogram of representative pixels from the  $F$  and  $B$  scribbles. Some pixels in the background fit the foreground color model much better than the background one (one such pixel is marked red in 8(b) and indicated by an arrow in 8(d)). As a result such pixels are classified as foreground with a high degree of certainty in the first stage. Once this error has been made it only reinforces further erroneous decisions in the vicinity of that pixel, resulting in a white clump in the alpha matte.

Since our method does not make use of global color models for  $F$  and  $B$  it can handle ambiguous situations such as that in Figure 8. However, there are also cases where our method fails to produce an accurate matte for the very same reason. Figure 9 shows an actress in front of a background with two colors. Even though the black  $B$  scribbles cover both colors the generated matte includes parts of the background (between the hair and the shoulder on the left). In such cases, the user would have to add another  $B$  scribble in that area.

## 6. Discussion

Matting and compositing are tasks of central importance in image and video editing and pose a significant challenge for computer vision. While this process by definition re-

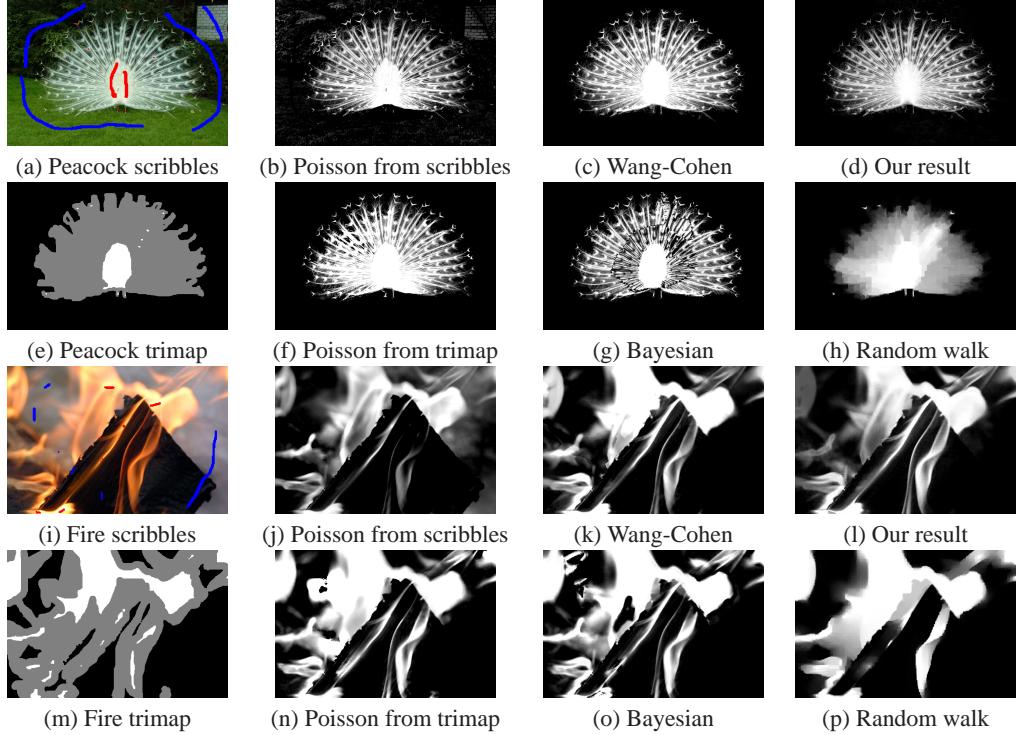


Figure 5. A comparison of alpha mattes extracted by different algorithms. Images (a,c,e,g,i,k,m,o) are taken from [15]. The remaining images were generated by our own implementation of the respective methods.

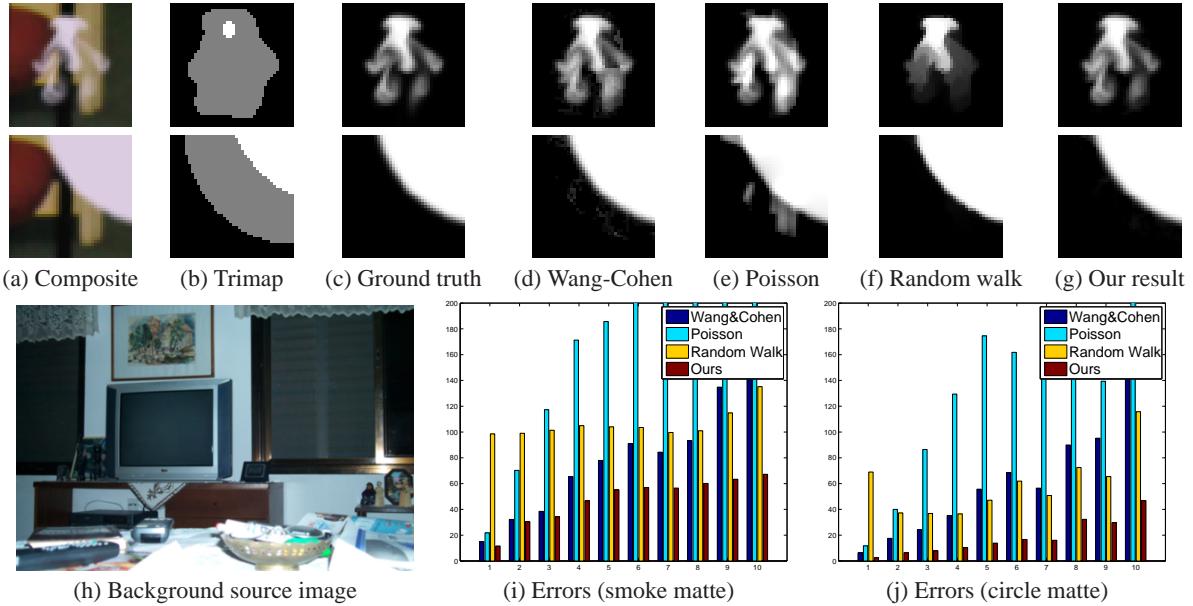


Figure 6. A quantitative comparison using two ground truth mattes. The errors in (i) and (j) are plotted as a function of average gradient strength of the background, binned into 10 bins. To produce these results we used our own implementation of the respective methods, using the parameter values specified in the original papers.

quires user interaction, the performance of most existing algorithms deteriorates rapidly as the amount of user input decreases. In this paper, we have introduced a cost function based on the assumption that foreground and background

colors vary smoothly and showed how to analytically eliminate the foreground and background colors to obtain a quadratic cost function in alpha. The resulting cost function is similar to cost functions obtained in spectral meth-

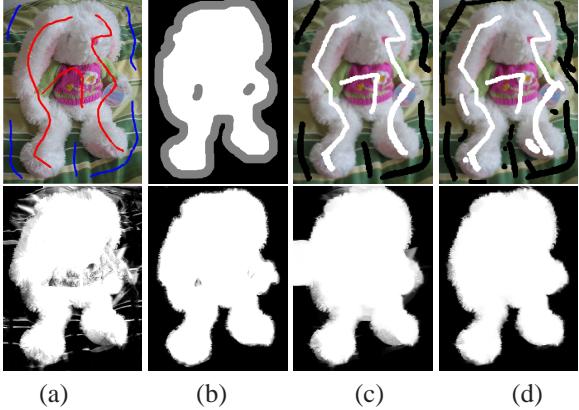
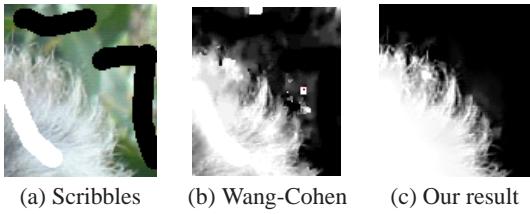


Figure 7. An example (from [15]) with color ambiguity between foreground and background. (a) scribbles and matte by [15]; (b) [15] results using a trimap; (c) our result with scribbles similar to those in (a); (d) our results with a few additional scribbles.



(d) RGB histogram of  $F$  (red) and  $B$  (blue) pixels.

Figure 8. An example with ambiguity between  $F$  and  $B$ .

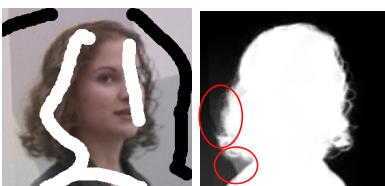


Figure 9. Failure due to lack of a color model.

ods to image segmentation but with a novel affinity function that is derived from the formulation of the matting problem. The global minimum of our cost can be found efficiently by solving a sparse set of linear equations. Our experiments on real and synthetic images show that our algorithm clearly outperforms other algorithms that use quadratic cost functions which are not derived from the matting equations. Our experiments also demonstrate that our results are competitive with those obtained by much more complicated, non-linear, cost functions. However, compared to previous non-linear approaches, we can obtain solutions in a few seconds,

and we can analytically prove properties of our solution and provide guidance to the user by analyzing the eigenvectors of our operator.

While our approach assumes smoothness in foreground and background colors, it does not assume a global color distribution for each segment. Our experiments have demonstrated that our local smoothness assumption often holds for natural images. Nevertheless, it would be interesting to extend our formulation to include additional assumptions on the two segments (e.g., global models, local texture models, etc.). The goal is to incorporate more sophisticated models of foreground and background but still obtain high quality results using simple numerical linear algebra.

## References

- [1] N. E. Apostoloff and A. W. Fitzgibbon. Bayesian video matting using learnt image priors. In *Proc. CVPR*, 2004.
- [2] A. Berman, P. Vlahos, and A. Dadourian. Comprehensive method for removing from an image the background surrounding a selected object. US Patent no. 6,135,345, 2000.
- [3] Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *Proc. ICCV*, 2001.
- [4] Y. Chuang, A. Agarwala, B. Curless, D. Salesin, and R. Szeliski. Video matting of complex scenes. *ACM Trans. Graph.*, 21(3):243–248, 2002.
- [5] Y. Chuang, B. Curless, D. Salesin, and R. Szeliski. A Bayesian approach to digital matting. In *Proc. CVPR*, 2001.
- [6] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann. Random walks for interactive alpha-matting. In *Proc. VIIP05*.
- [7] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 2004.
- [8] A. Levin, D. Lischinski, and Y. Weiss. A closed form solution to natural image matting. Hebrew University Technical Report, 2006.
- [9] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Trans. Graph.*, 23(3):303–308, 2004.
- [10] I. Omer and M. Werman. Color lines: Image specific color representation. In *Proc. CVPR 2004*, June 2004.
- [11] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004.
- [12] M. Ruzon and C. Tomasi. Alpha estimation in natural images. In *Proc. CVPR*, 2000.
- [13] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proc. CVPR*, pages 731–737, 1997.
- [14] J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum. Poisson matting. *ACM Trans. Graph.*, 23(3):315–321, 2004.
- [15] J. Wang and M. Cohen. An iterative optimization approach for unified image segmentation and matting. In *Proc. IEEE Int'l. Conf. on Computer Vision*, 2005.
- [16] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems 17*. 2005.
- [17] A. Zomet and S. Peleg. Multi-sensor super resolution. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, 2002.