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

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 for natural images may be obtained from a small amount of user input.

Keywords: Matting, Interactive Image Editing, Spectral Segmentation

I. 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

(a) (b) (c) (d) (e) (f)

Fig. 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 [11], [13] 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 [19])

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,

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

where α_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 α simultaneously. This is typically done by iterative nonlinear optimization, alternating the estimation of F and B with that of α. 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 [19]. 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 α. 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 α directly and without requiring reliable estimates for F and B, a modest 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.

A. Previous work

Most existing methods for natural image matting require the input image to be accompanied by a *trimap* [1], [2], [5], [6], [14], [17], 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 [6], [14] 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 [6], where a mixture of oriented Gaussians is used to learn the local distribution and then α, 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 [17], 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], [11], [13]. 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. [13] 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. [10], and the random walk alpha matting method of Grady et al. [8]. 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.

A scribble-based interface for interactive matting was proposed by Wang and Cohen [19]. 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. Another scribble-based matting approach was recently proposed by Guan et al. [9], augmenting the random walk approach [8] with an iterative estimation of color models.

II. 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 II-A.

As mentioned earlier, the matting problem is severely under-constrained. Therefore, some assumptions on the nature of F, B and/or α 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 α can account for the discontinuities in I. This assumption, which will be somewhat relaxed in Section II-A, allows us to rewrite (1) expressing α as a linear function of the image I:

$$\alpha_i \approx aI_i + b, \quad \forall i \in w, \tag{2}$$

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 [20], and the shape recipes of [18]. This relation suggests finding α, a and b that minimize the cost function

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

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 α mattes (since $a_j = 0$ means that α is constant over the j-th window).

In our implementation, we typically use windows of 3×3 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 α, 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

$$J(\alpha) = \alpha^T L\, \alpha, \tag{4}$$

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

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

Here δ_{ij} is the Kronecker delta, μ_k and σ_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

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

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 α we define $\bar{\alpha}_k$ as a $(|w_k|+1) \times 1$ vector, whose entries are α_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 α the optimal pair a_k^*, b_k^* inside each window w_k is the solution to the least squares problem:

$$(a_k^*, b_k^*) = \operatorname{argmin} \left\| G_k \begin{bmatrix} a_k \\ b_k \end{bmatrix} - \bar{\alpha}_k \right\|^2 \tag{7}$$

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

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

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

and some further algebraic manipulations show that the (i,j)-th element of $\bar{G}_k^T \bar{G}_k$ 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). □

A. 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:

$$\alpha_i \approx \sum_c a^c I_i^c \ + b, \quad \forall i \in w, \tag{9}$$

where c sums over color channels. 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. In Figure 2 we illustrate this concept, by plotting local RGB distributions from a real image. Furthermore, Omer and Werman [12] 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 3×3 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

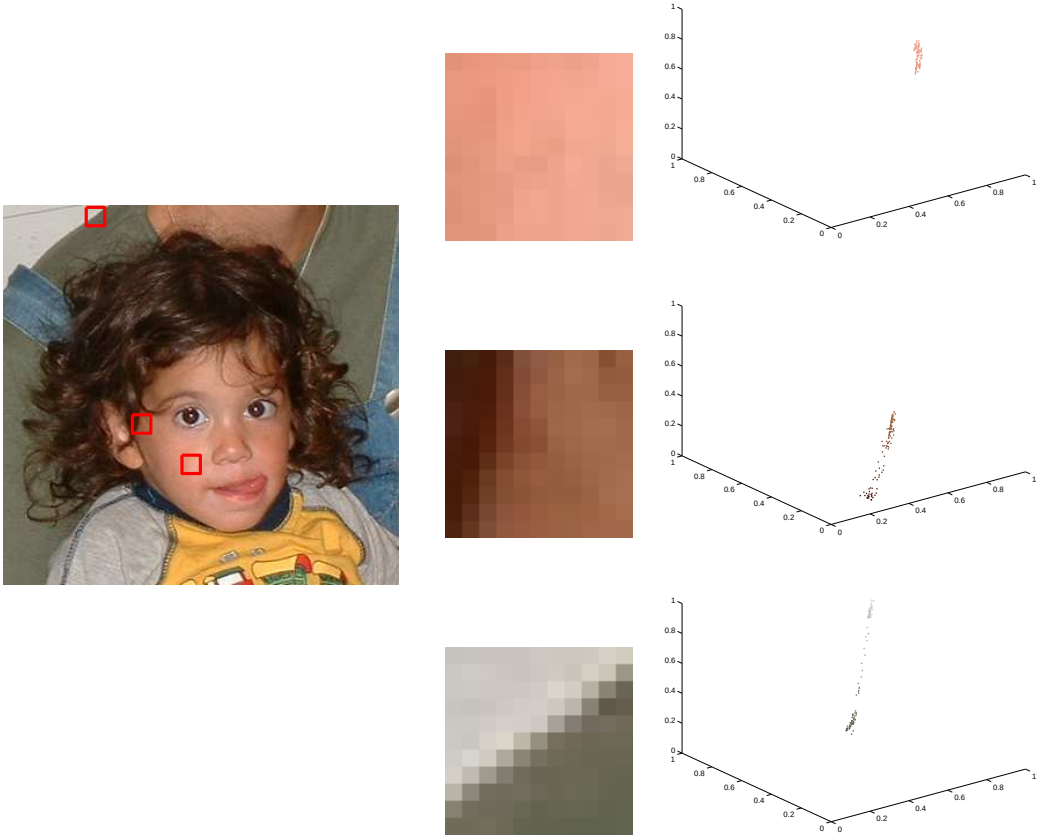


Fig. 2. Local patches selected from a real image and the RGB plots of their color distributions.

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 3×1 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 $\alpha_i = \sum_c a^c I_i \ + b$. □

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

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

Similarly to the grayscale case, a^c and b can be eliminated from the cost function, yielding a

quadratic cost in the α unknowns alone:

$$J(\alpha) = \alpha^T L\, \alpha. \tag{11}$$

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

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

where Σ_k is a 3×3 covariance matrix, μ_k is a 3×1 mean vector of the colors in a window w_k, and I_3 is the 3×3 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 (as each of the color channels can be expressed as a linear function of itself, e.g., by setting $a^1 = 1, a^2 = a^3 = b = 0$).

Apart from the mathematical justification, the intuition behind our cost function is that the matte may be represented locally as a linear combination of the image color channels, as illustrated by the three representative examples shown in Figure 3. The first example is a window with rather uniform foreground and background colors. In this case the alpha matte has a strong normalized correlation with the image and it may be generated by multiplying one of the color channels by a scale factor and adding a constant. In the second example, the alpha matte is constant over the entire window. Regardless of the complexity of the image texture in this window, we can obtain the constant alpha by multiplying the image channels by zero and adding a constant. This trivial case is important, as it demonstrates some of the power of the 4D linear model. Since a typical matte is constant (0 or 1) over most image windows, the matte in such windows may be expressed as a linear function of the image in a trivial way, regardless of the exact color distribution, and whether the color line model holds or not. Finally, we present a window with non-uniform alpha, where, in addition, the background contains an edge. Since the edge contrasts in the different color channels are different, by scaling the color channels appropriately our model is able to actually cancel the background edge.

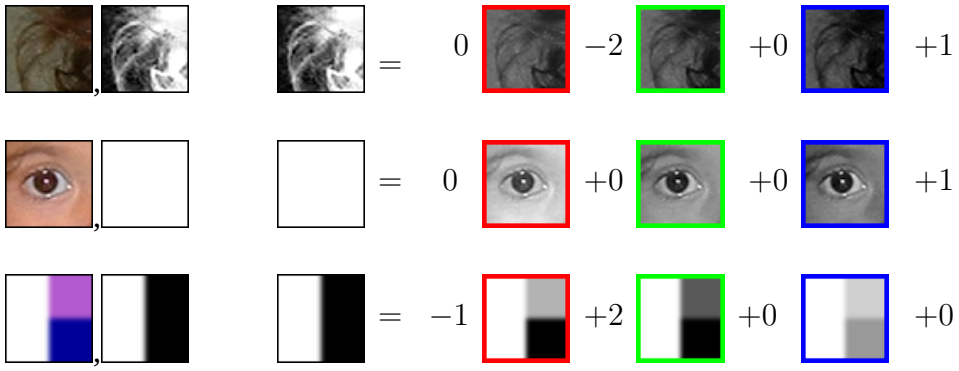


Image and matte pair Expressing the matte as a linear combination of color channels

Fig. 3. Local linear relations between alpha windows and image windows.

III. Constraints and User Interface

In our system the user-supplied constraints on the matte may be provided via a scribble-based GUI, or a trimap. 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 constraints we solve for

$$\alpha = \text{argmin } \alpha^T L\, \alpha + \lambda(\alpha^T - b_S^T) D_S\, (\alpha - b_S) \tag{13}$$

where λ is some large number, D_S is a diagonal matrix whose diagonal elements are 1 for constrained pixels and 0 for all other pixels, and b_S is the vector containing the specified alpha values for the constrained pixels and 0 for all other pixels.

Since the above cost is quadratic in alpha, the global minimum may be found by differentiating (13) and setting the derivatives to 0. This amounts to solving the following sparse linear system:

$$(L + \lambda D_S)\alpha = \lambda b_S \tag{14}$$

Theorem 3: Let I be an image formed from F and B according to the compositing equation (1), and let α^* 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 α^*, then α^* 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 $J(\alpha^*) = \alpha^{*T} L\, \alpha^* = 0$. □

We demonstrate this in Figure 4. The first image (Figure 4(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 4(b)) is indeed *identical* to the ground truth matte. The second example (Figure 4(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 4(d)).

Note that theorem 3 states only necessary but not sufficient conditions for recovering the true alpha matte. This is the case, since the nullspace of L may contain multiple solutions and it is up to the user to provide a sufficient number of constraints to ensure that solving equation (13) yields the correct alpha matte. For example, constraining the system to output the true matte in Figure 4(a), required a setting constraint inside every connected component.

(a) (b) (c) (d)

Fig. 4. Matting examples. (a,c) Input images with sparse constraints. (b,d) Extracted mattes.

A. Additional Scribbling Brushes

To provide the user with more flexible control over the output we add additional types of brushes for specifying constraints at regions containing mixed pixels. One simple constraint may be set by explicitly specifying the values of F and B under the scribble (by cloning them from other locations in the image). This gives a constraint on α in the scribbled area, computable directly from the compositing equation (1). Another constraint type is when the artist indicates that F and B are *constant but unknown* under the scribble. This tells the system that the linear

relationship (9) should hold for all the pixels covered by the scribble, rather than only inside each 3×3 window. This adds to equation (12) an additional larger window which contains all of the pixels under the scribble. The usage of these two brushes is illustrated in Figure 5. If only black and white scribbles are used (Figure 5(a)), there is no way to specify background constraints for the parts of the background partially visible through the fine gaps in the hair. As a result these gaps are not captured by the recovered matte (Figure 5(b)). To overcome this, the user may pick some blond and brown colors from the neighboring pixels (Figure 5(c)) and specify them as the foreground and background colors in that region. The matte produced from these constraints succeeds in capturing the partial visibility of the background. Alternatively, the user may place a scribble (gray scribble in Figure 5(d)) indicating that this area should be treated as a single large neighborhood, causing the brown pixels showing through the hair to be treated the same as the brown pixels outside. In the results section we show that these additional brushes are also useful for the challenging tasks of extracting mattes for shadows and smoke.

IV. PARAMETERS

To gain a better understanding of our method, we illustrate here the effect of the different parameters on the alpha reconstruction.

First we demonstrate the effect of ϵ, which is the weight of the regularization term on a in eq. (3). There are two reasons for having this term. The first reason 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 α mattes (since $a_j = 0$ means that α is constant over the j-th window). In Figure 6 we demonstrate the effect of ϵ on the resulting matte. Our input image consists of two noisy areas, and was scribbled with two vertical lines. In Figure 6 we show three different mattes that were obtained using three different values of ϵ. We also plot the different mattes using a one-dimensional profile of one of the rows. For comparison we also plot the profile of the input image scaled to the $[0, 1]$ range. We can see that when ϵ is small the sharpness of the recovered matte matches the profile of the edge in the input image, but the matte also captures the image noise. For large ϵ values the image noise is indeed ignored, but the recovered alpha is over-smoothed. In our implementation we usually used $\epsilon = 0.1^7$ to 0.1^5, since real images are normally not as noisy as the above example.

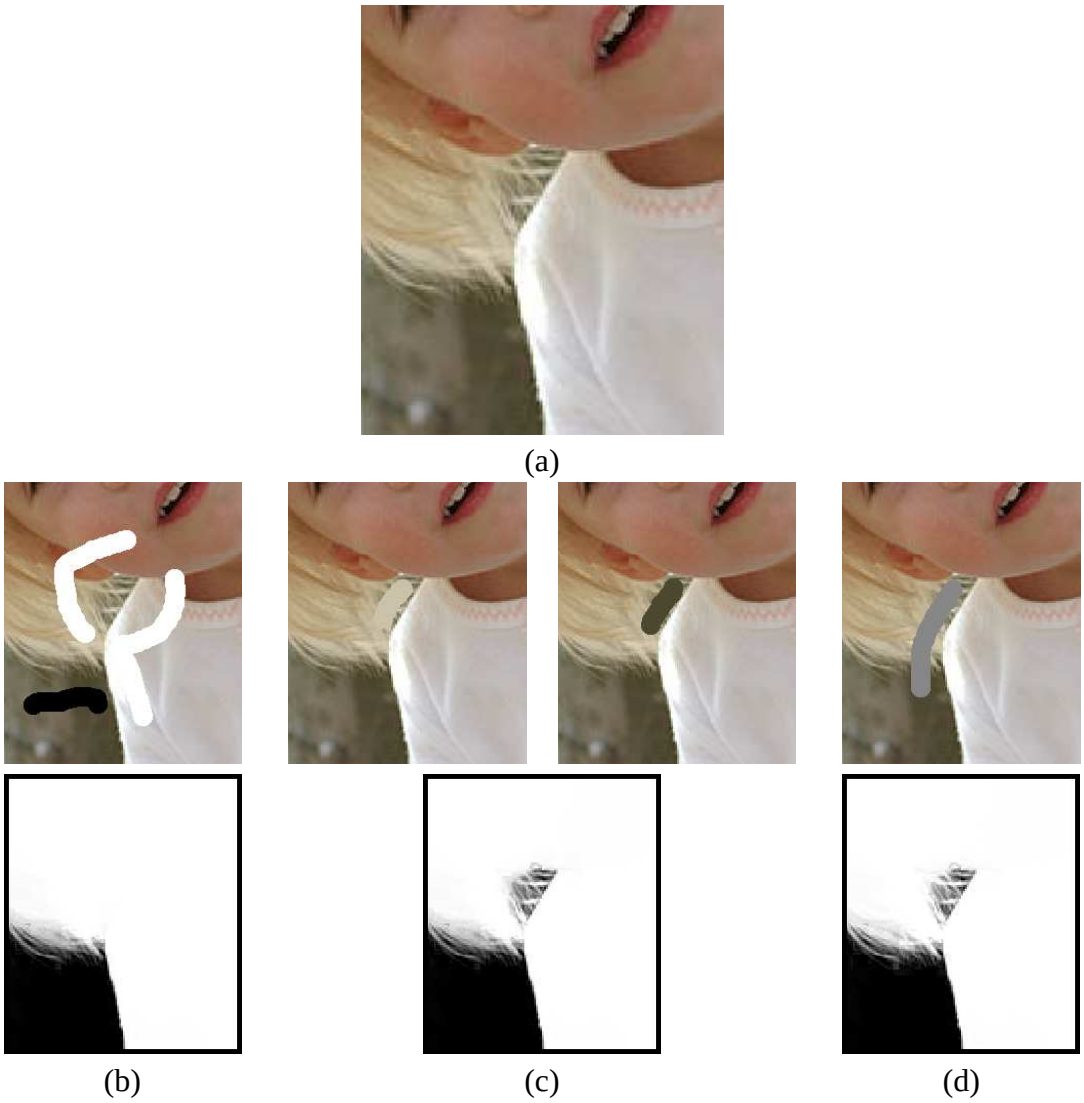


Fig. 5. Using additional scribbling brushes. (a) Input image. (b) Simple background (black) and foreground (white) scribbles. (c) Scribbling foreground and background colors explicitly. (d) Marking wider neighborhoods.

Figure 6 demonstrates the fact that ϵ is an important parameter in our system, which controls the amount of noise versus the amount of smoothing in the solution. While many of our theoretical results in this paper only hold for the case $\epsilon = 0$, it should be noted that, in practice, as ϵ approaches 0, our method will typically fail to produce a constant matte in textured or noisy regions.

Another parameter which affects the results is the window size. We usually construct the matting Laplacian using 3×3 windows. Using wider windows is more stable when the color lines model holds, but the chance of encountering windows that deviate from the color lines

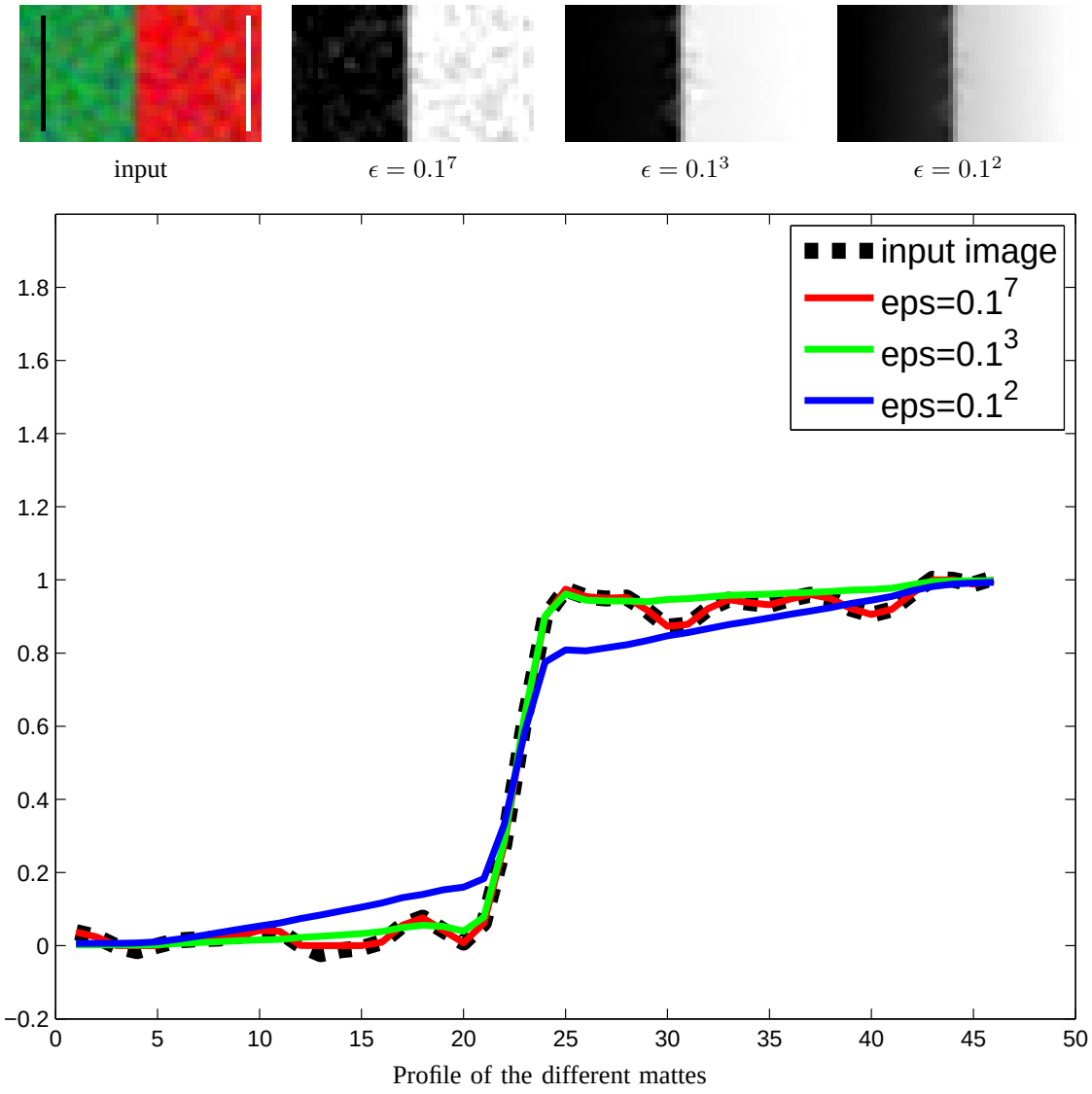
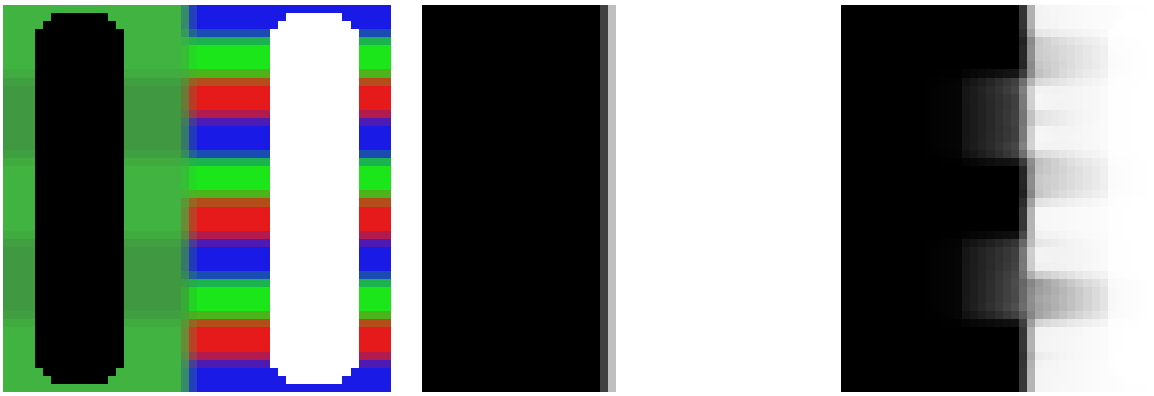


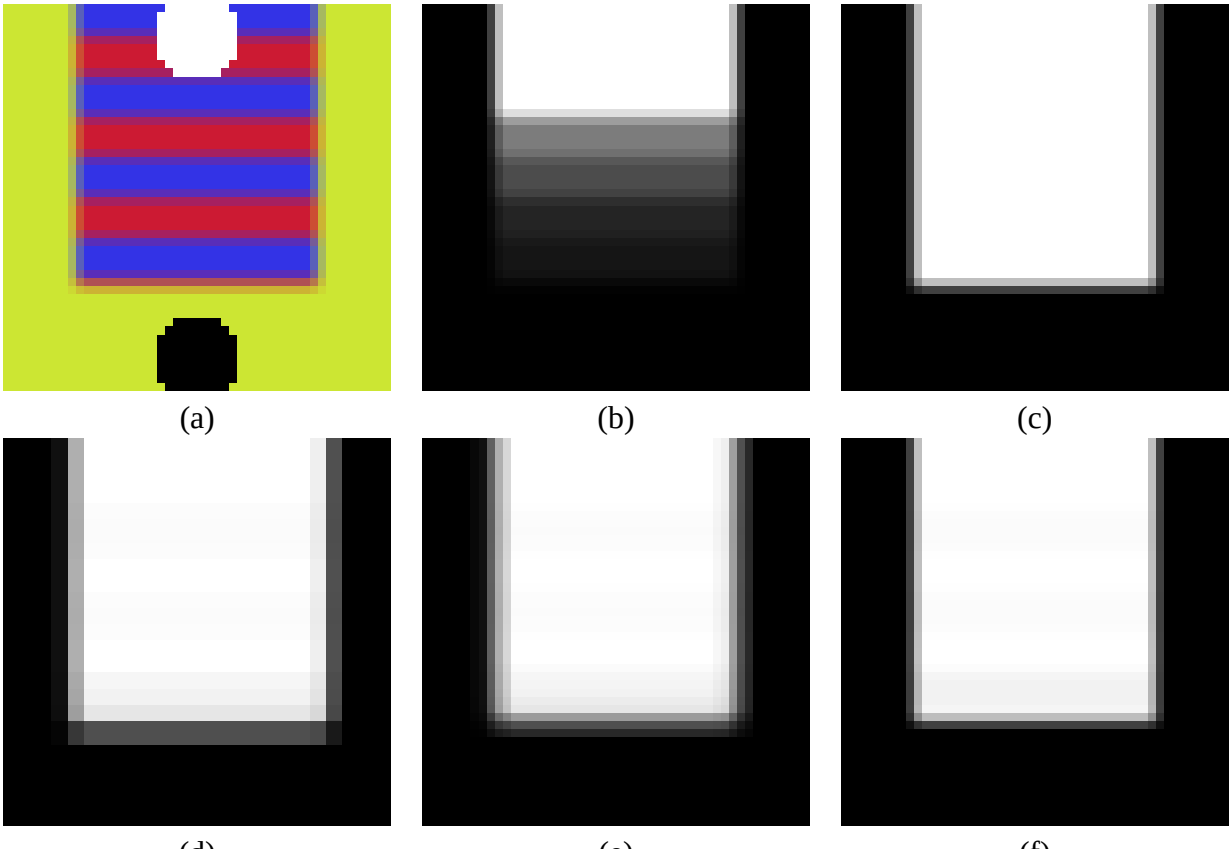
Fig. 6. Computing a matte using different ϵ values.

model grows when the windows are larger. This is illustrated in Figure 7. The matte of the image was recovered using both 3×3 and 5×5 windows. The mattes are shown in Figure 7(b) and (c), respectively. It may be seen that the matte in Figure 7(c) contains some errors. The reason is that some of the 5×5 windows deviate from the color lines model since their areas cover three differently colored background strips, while the 3×3 windows are small enough and never cover more then two strips. On the other hand, in Figure 8 the fact that 5×5 windows can cover three different strips is useful as that helps the foreground constraint (the white scribble)



(a) input marks (b) 3×3 windows (c) 5×5 windows

Fig. 7. Computing a matte using different window sizes.



(a) (b) (c)

(d) (e) (f)

Fig. 8. Computing a matte using different window sizes. (a) Input marks. (b) 3×3 windows. (c) 5×5 windows. (d) 3×3 windows computed at coarser resolution. (e) Simple interpolation of (d). (f) Interpolating the a, b parameters corresponding to the matte in (d) and applying them to obtain a matte for the finer image.

to propagate to the entire striped texture region (Figure 8(c)). (Despite the fact that two blue

pixels in different strips are not direct neighbors, they are neighbors in the induced graph due to the fact that the window is large enough). Such propagation does not occur when using 3×3 windows, as shown in Figure 8(b).

Even in cases for which wider windows are useful, their usage increases computation time since the resulting system is less sparse. To overcome this, we consider the linear coefficients in equation (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 3×3 windows on a coarse image and compute the linear coefficients that relate it to the coarse image channels. We then interpolate the linear coefficients and apply them to the finer resolution image. The alpha matte obtained using this approach is similar to the one that would have been obtained by solving the matting system directly on the fine image with wider windows. To demonstrate this, Figure 8(d) shows the alpha matte that was obtained when 3×3 windows where used on the image of Figure 8(a) after downsampling it by a factor of 2. If we just upsample this alpha matte by a factor of two we get the blurred alpha matte shown in Figure 8(e). On the other hand, if we compute the a, b values relating the small alpha (Figure 8(d)) to the image, upsample them and apply them to the finer resolution image, we get the sharp matte in Figure 8(f), which is almost identical to the one in Figure 8(c), obtained using 5×5 windows.

V. Spectral Analysis

The matting Laplacian matrix L is a symmetric semi-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 has the same form as 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., for image segmentation using normalized cuts [15]) is to set

$$W_{\mathrm{G}}(i,j) = e^{-\|I_i - I_j\|^2/\sigma^2}, \tag{15}$$

where σ 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 σ. The

random walk matting algorithm [8] uses a similar affinity function for the matting problem, but the color distance between two pixels is taken after applying a linear transformation to their colors. The transformation is image-dependent and is estimated using a manifold learning technique.

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":

$$W_{\text{M}}(i,j) = \sum_{k|(i,j)\in w_k} \frac{1}{|w_k|}(1 + (I_i - \mu_k)(\Sigma_k + \frac{\epsilon}{|w_k|}I_3)^{-1}(I_j - \mu_k)) \tag{16}$$

We note that by using the term *affinity* here, we somewhat extend its conventional usage: while standard affinities are usually non negative, the matting affinity may also assume negative values.

To compare the two affinity functions W_{G} and W_{M} we examine the eigenvectors of the corresponding Laplacians, since these eigenvectors are used by spectral segmentation algorithms for partitioning images.

Figure 9 shows the second smallest eigenvector (the first smallest eigenvector is constant in both cases) for both Laplacian matrices, on three example images. For the matting affinity, we present eigenvectors with two ϵ values ($\epsilon = 0.1^7$ and $\epsilon = 0.1^5$). The first example is a simple image with concentric circles of different color. In this case the boundaries between regions are very simple, and all Laplacians capture the transitions correctly. The second example is an image of a peacock. The global σ eigenvector (used by standard spectral clustering algorithms) fails to capture the complex fuzzy boundaries between the peacock's tail feathers and the background. In contrast, the matting Laplacian's eigenvector (constructed using $\epsilon = 0.1^5$) separates the peacock from the background very well, as this Laplacian explicitly encodes fuzzy cluster assignments. When the matting Laplacian is constructed using $\epsilon = 0.1^7$ the eigenvector is similar to the input image and in addition to the peacock also captures some of the vegetation in the background. The last example is the noisy step function from Figure 6. In this case, the eigenvector corresponding to $\epsilon = 0.1^7$ captures all of the image noise, while using a larger ϵ results in a less noisy eigenvector. However, an appropriate choice of a global σ yields an eigenvector with a perfect step function. This is an excellent result if the goal is a hard segmentation, but if the goal is a soft alpha matte, it is preferable to have an edge whose smoothness is proportional to the smoothness of the edge in the input image, so the matting eigenvector might be more appropriate.

Thus, designing a good matting affinity is not equivalent to designing a good affinity for hard segmentation.

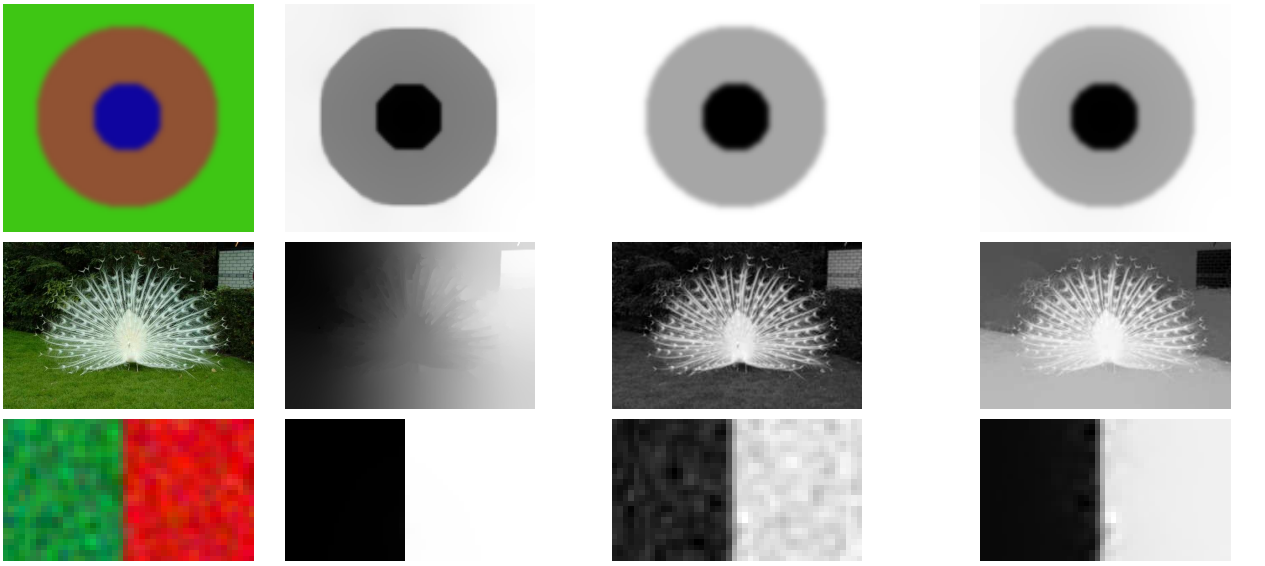


Fig. 9. Smallest eigenvectors of Laplacians corresponding to different affinity functions.

A. The eigenvectors as guides

While the matting problem is ill-posed without some user input, the matting Laplacian matrix contains a lot of information on the image even before any constraints have been provided, as demonstrated in the previous section.

This suggests that looking at some of the smallest eigenvectors of the matting Laplacian can guide the user *where to place scribbles*. For example, the extracted matte and the smallest eigenvectors tend to be piecewise constant over the same regions. 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. We note, however, that a basic strategy for scribble placing is just to examine the input image and place scribbles on regions with different colors. This is also evident by the fact that when the matting Laplacian is constructed using $\epsilon = 0$, the nullspace of the matting Laplacian will contain the 3 color channels.

Figure 10 illustrates how a scribbling process may be guided by the eigenvectors. By examining the two smallest eigenvectors (Figure 10(a-b)) we placed a scribble inside each region exhibiting coherent eigenvector values (Figure 10(c)). The resulting matte is shown in Figure 10(d). Note that the scribbles in Figure 10(c) were our first, and single attempt to place scribbles on this image.

Stated somewhat more precisely, the alpha matte may be predicted by examining some of 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, it is possible to bound the weight of the larger eigenvectors in the optimal solution, as a function of the ratios of the corresponding eigenvalues.

(a) (b) (c) (d)

Fig. 10. Smallest eigenvectors (a-b) are used for guiding scribble placement (c). The resulting matte is shown in (d).

Theorem 4: Let $v_1, \ldots, v_N$ be the eigenvectors of the matting Laplacian (12) with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$. Let S be the subset of scribbled pixels, with scribble values $s_i, \quad i \in S$. We denote by $x(S)$ the restriction of the vector x to the scribbled pixels (so that $x(S)$ is an $|S|$ dimensional vector). Let α be the optimal matte and suppose α is expressed with respect to the eigenvectors basis as $\alpha = \sum_{k=1}^{N} a_k v_k$.

If the scribbles are spanned by the K smallest eigenvectors $s(S) = \sum_{k=1}^{K} b_k v_k(S)$, then for every $j > K$:

$$a_j^2 \leq \frac{\sum_{k=1}^{K} b_k^2}{\lambda_j} \leq \frac{\|b\|^2 \lambda_K}{\lambda_j}$$

Proof: Let $\beta = \sum_{k=1}^{K} b_k v_k$. Then β satisfies $\beta(S) = s(S)$. Since α is the optimal solution $\alpha = \operatorname{argmin} \ \alpha^T L \ \alpha$, s.t. $\alpha(S) = s(S)$, we must have that $\alpha^T L \alpha \leq \beta^T L \beta$. Since the Laplacian

matrix L is positive semi-definite, the eigenvectors $v_1, \ldots, v_N$ are orthogonal. Therefore,

$$\alpha^T L \alpha = \sum_{k=1}^{N} a_k^2 \lambda_k, \tag{17}$$

$$\beta^T L \beta = \sum_{k=1}^{K} b_k^2 \lambda_k, \tag{18}$$

and as a result for every j: $a_j^2 \lambda_j \leq \sum_{k=1}^{K} b_k^2 \leq \|b\|^2 \lambda_K$. □

Corollary 1: If the scribbles are spanned by the nullspace of L the optimal solution will also lie in the nullspace of L.

Proof: Let K be the dimension of the nullspace. Using the previous theorem's notation, for every $j > K$, $a_j^2 \leq \|b\|^2 \lambda_K = 0$, and the optimal solution is spanned by the K nullspace eigenvectors. □

The above implies that the smoothness of the recovered alpha matte will tend to be similar to that of the smallest eigenvectors of L.

VI. Optimization

The optimization problem defined by equation (13) is one of minimizing a quadratic cost function subject to linear constraints, and the solution can therefore be found by solving a sparse set of linear equations.

For the results shown here we solve the linear 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 large images using Matlab's solver is impossible due to memory limitations. To overcome this we use a coarse-to-fine scheme. We downsample the image and the constraints and solve at a lower resolution. The recovered alpha matte is then interpolated to the finer resolution, alpha values are thresholded and pixels with alpha close to 0 or 1 are clamped and considered as constrained in the finer resolution. Constrained pixels may be eliminated from the system, reducing the system size. For that, we note that within the constrained areas there is no need to enforce the local linear models. Therefore, when computing the matting Laplacian matrix (equations 5,12), we sum only windows w_k that contain at least one unconstrained pixel. Aside for efficiency, clamping alpha values to 0 or 1 is also useful in avoiding over-smoothed α-mattes, and we used such clamping to produce the results in Figure 16(d). We note, however,

that clamping the alpha values in a coarse to fine approach has the negative side effect that long thin structures, such as strains of hair, may be lost.

It should be noted that solving linear systems with this structure is a well studied problem [8], [16]. We have also implemented a multigrid solver for matte extraction. The multigrid solver runs in a couple of seconds even on very large images, but with a small degradation in matte quality. Therefore, a multigrid solver enables the system to operate as an interactive tool. The user can place constraints, examine the resulting matte and add constraints in image areas which require further refinement.

The eigenvectors of the matting Laplacian depend only on the input image, and are independent of the user's constraints. It is not necessary to compute them, unless the user wishes to use them for guidance in scribble placement, as described earlier. In this case, they only need to be computed once, possibly as part of the initialization that takes place when a new image is loaded. Recently, there has been much research of efficient methods for computation of eigenvectors (e.g., [4]), partly in response to the growing interest in normalized cuts image segmentation and other spectral clustering methods.

VII. Reconstructing F and B

Having solved for α it is also usually necessary to reconstruct F, and in some cases also B. One approach for reconstructing F and B is to solve equation (9) for the optimal a, b given α using least squares. However, in order to extract F and B from a, b there is an additional matting parameter that should be recovered (β in the proof of theorem 2). For complex foreground and background patterns such a reconstruction may produce noisy results, and therefore we solve for F and B using the compositing equation, introducing some explicit smoothness priors on F and B. The smoothness priors are stronger in the presence of matte edges. Specifically, we minimize a system of the form:

$$\begin{aligned} min \sum_{i \in I} \sum_{c} & \quad (\alpha_i F_i^c + (1-\alpha_i) B_i^c - I_i^c)^2 \\ + & \quad |\alpha_{i_x}| \left((F_{i_x}^c)^2 + (B_{i_x}^c)^2 \right) + |\alpha_{i_y}| \left((F_{i_y}^c)^2 + (B_{i_y}^c)^2 \right) \end{aligned} \tag{19}$$

where $F_{i_x}^c$, $F_{i_y}^c$, $B_{i_x}^c$, and $B_{i_y}^c$ are the x and y derivatives of F^c and B^c, and $\alpha_{i_x}, \alpha_{i_y}$ are the matte derivatives. We note that for a fixed α the cost (19) is quadratic and its minimum may be found by solving a sparse set of linear equations. Given the solution of F and B the α solution

can be further refined, but in practice we have observed this is not required. Figure 11 shows an α-matte and F and B images recovered in this way.

(a) (b) (c) (d) (e)

Fig. 11. Foreground and background reconstruction: (a) input (b) α-matte (c) foreground reconstruction (d) background reconstruction (e) F composited over a novel background.

VIII. Results

In all examples presented in this section the scribbles used in our algorithm are presented in the following format: black and white scribbles are used to indicate the first type of hard constraints on α. Red scribbles represent places in which foreground and background colors where explicitly specified. Finally, gray scribbles are used to represent the third type of constraint — requiring a and b to be constant (without specifying their exact value) within the scribbled area.

A. Visual Comparisons

Figure 12 presents matting results on images from the Bayesian matting work [6]. Our results appear visually comparable to those produced by Bayesian matting. While the Bayesian matting results use a trimap, each of our results was obtained using a sparse set of scribbles.

In Figure 13 we extract mattes from a few of the more challenging examples presented in the Poisson matting paper [17]. For comparison, the Poisson and Bayesian matting results provided in [17] are also shown[1].

Figure 14 shows the mattes extracted using our technique on two challenging images used in [19] and compares our results to several other recent algorithms. It can be seen that our results on these examples are comparable in terms of visual quality to those of [19], even though we use a far simpler algorithm. Global Poisson matting cannot extract a good matte from a sparse

[1]We thank Leo Jia for providing us with the images and results

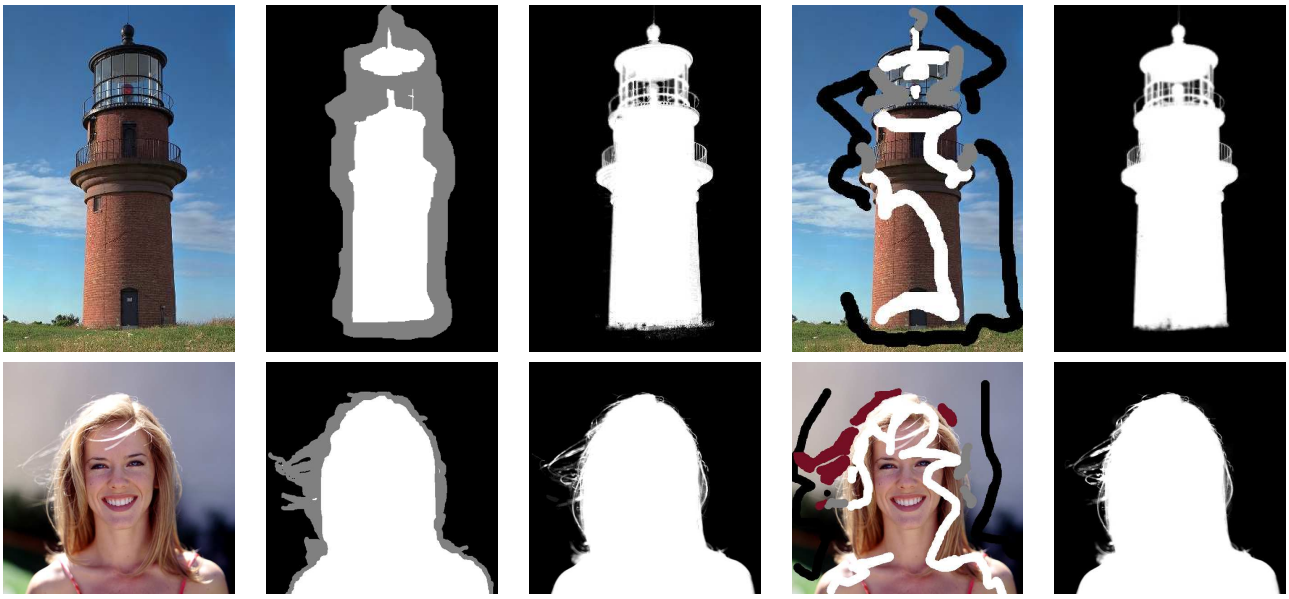


Fig. 12. Comparison with Bayesian matting [6]. (a) input image (b) trimap (c) Bayesian matting result (obtained from the Bayesian Matting webpage) (d) scribbles (e) our result.

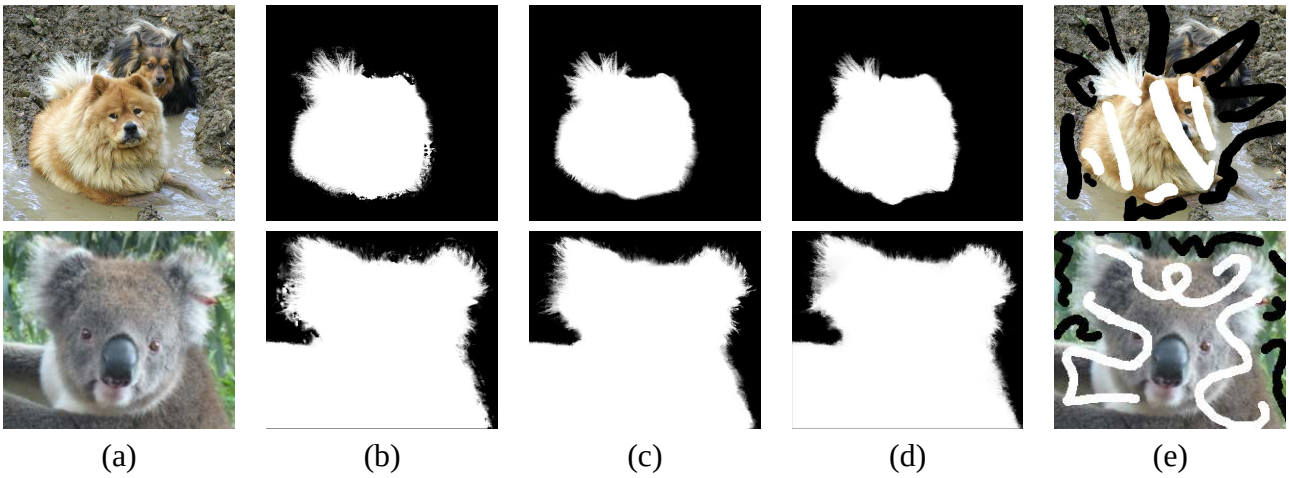
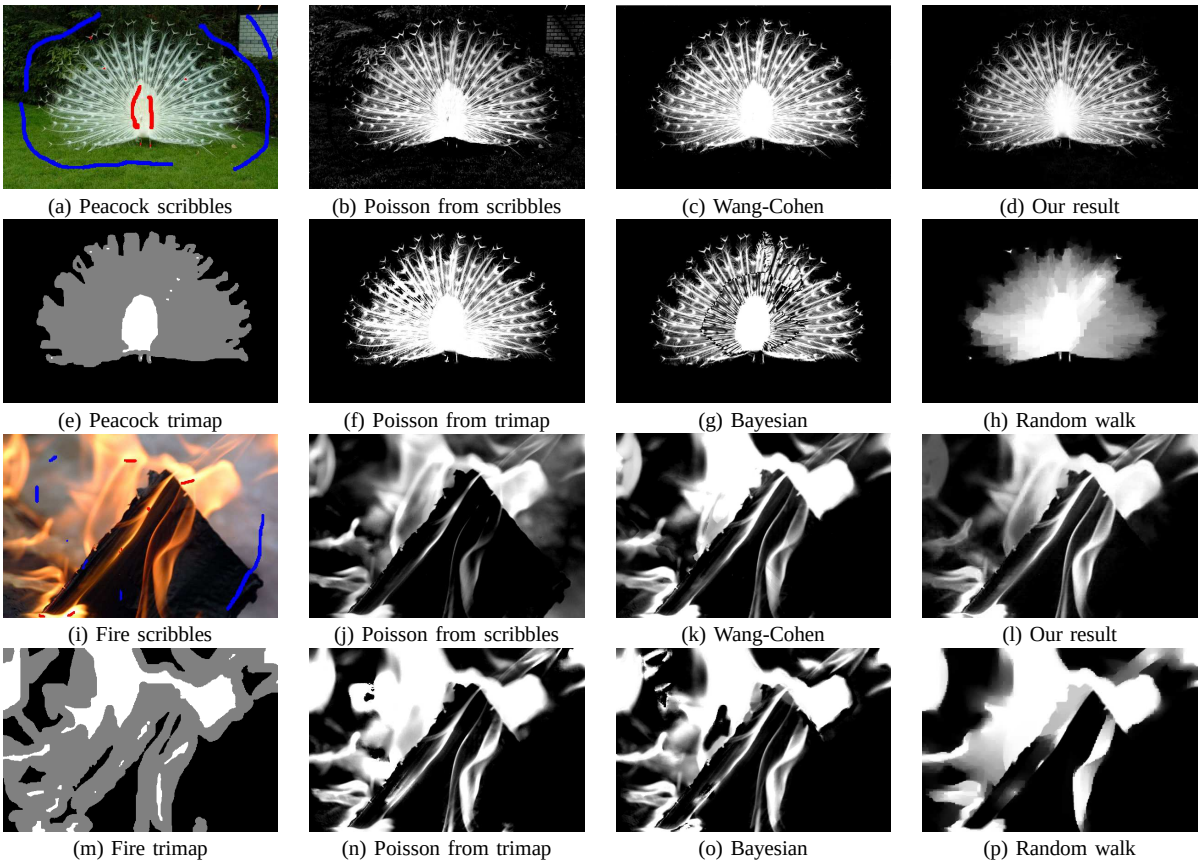


Fig. 13. Result on Poisson matting examples. (a) input image (b) Bayesian matting (obtained from the Poisson matting paper) (c) Poisson matting (obtained from the Poisson matting paper) (d) our result (e) scribbles

set of scribbles although its performance with a trimap is quite good. The random walk matting algorithm [8] also minimizes a Laplacian but uses an affinity function with a global scaling parameter and hence has a particular difficulty with the peacock image.

Figure 15 presents compositing examples using our algorithm for some images from the previous experiments. We show compositing both over a constant background and over natural



(a) Peacock scribbles (b) Poisson from scribbles (c) Wang-Cohen (d) Our result

(e) Peacock trimap (f) Poisson from trimap (g) Bayesian (h) Random walk

(i) Fire scribbles (j) Poisson from scribbles (k) Wang-Cohen (l) Our result

(m) Fire trimap (n) Poisson from trimap (o) Bayesian (p) Random walk

Fig. 14. A comparison of alpha mattes extracted by different algorithms. Images (a,c,e,g,i,k,m,o) are taken from [19]. The remaining images were generated by our own implementation of the respective methods.

images.

Figure 16 shows an example (from [19]), where Wang and Cohen's method fails to extract a good matte from sparse 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 imperfect matte from the same set of scribbles, but adding a small number of additional scribbles results in a better matte. (To produce this result, we applied clamping of alpha values as described in section VI.)

Figure 17 shows another example (a closeup of the Koala image from [17]), 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 then the

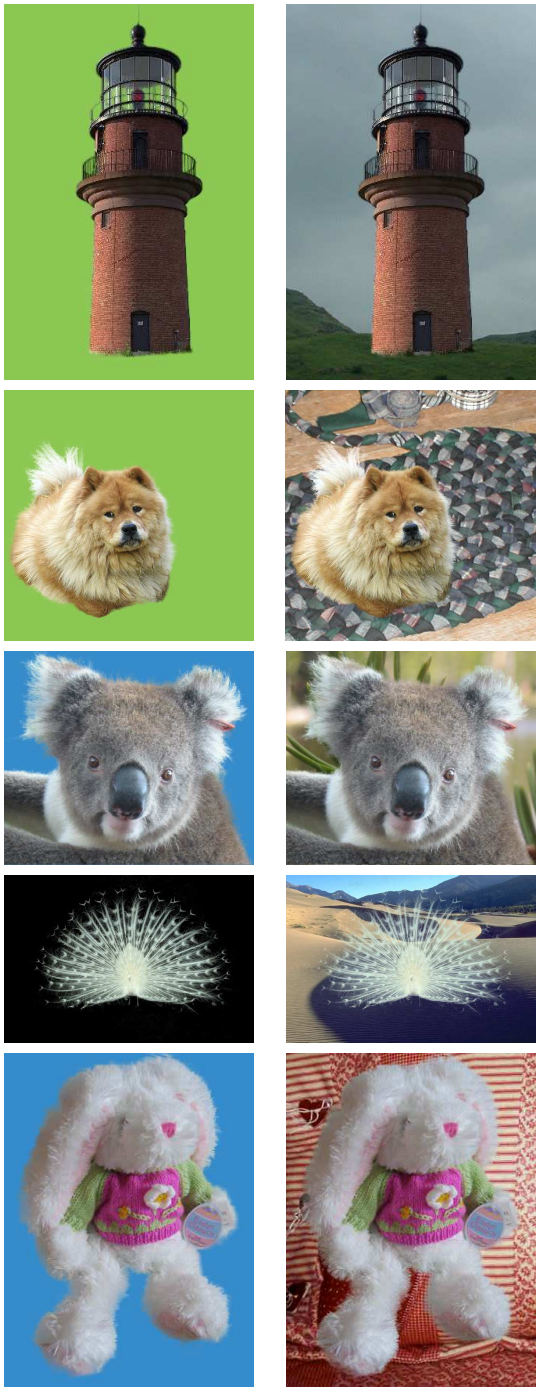
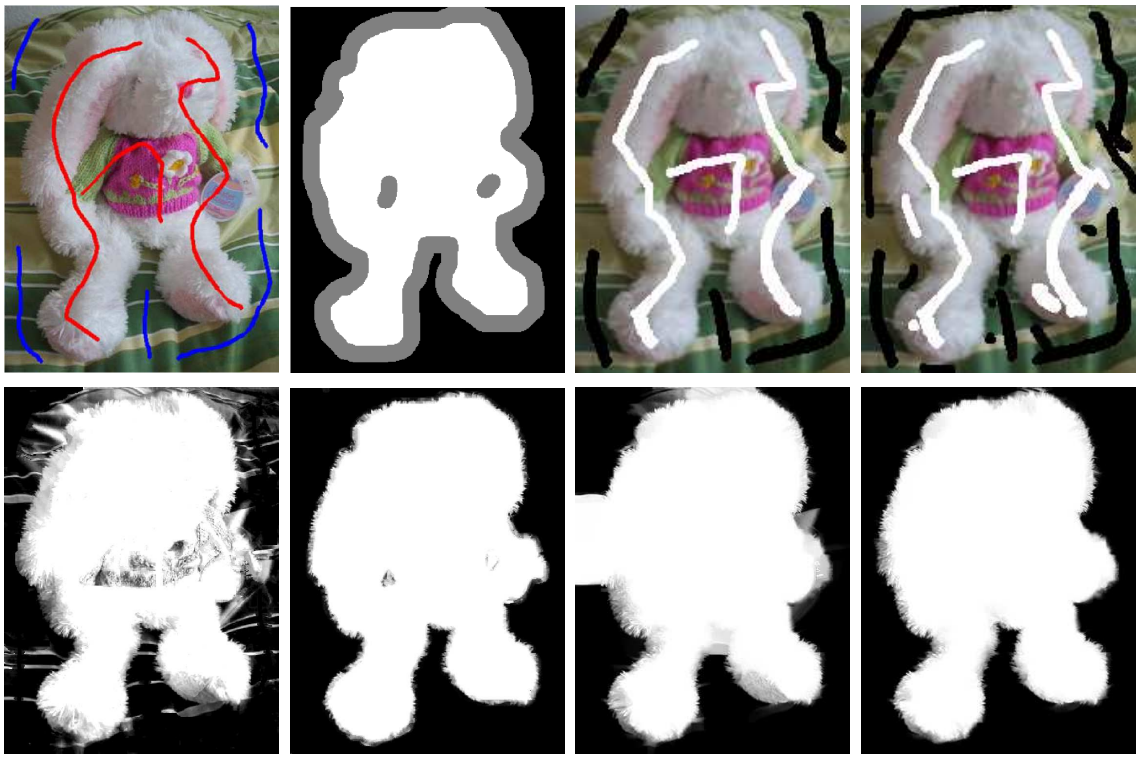


Fig. 15. Some compositing examples using α, F and B extracted by our algorithm. Left: compositing with a constant background. Right: compositing over natural images.

background one (one such pixel is marked red in 17(b) and indicated by an arrow in 17(d)). As a result such pixels are classified as foreground with a high degree of certainty in the first stage.



(a) (b) (c) (d)

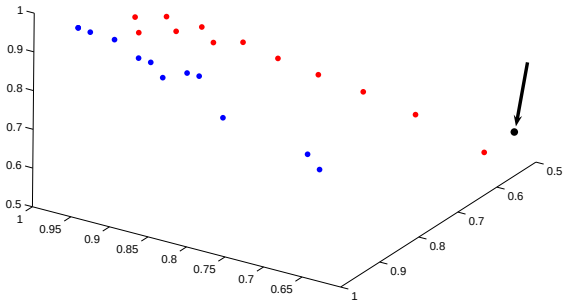
Fig. 16. An example (from [19]) with color ambiguity between foreground and background. (a) scribbles and matte by [19]; (b) [19] results using a trimap; (c) our result with scribbles similar to those in (a); (d) our results with a few additional scribbles.

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 17. However, there are also cases where our method fails to produce an accurate matte for the very same reason. Figure 18 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.

To demonstrate the limitations of our approach in the presence of insufficient user input, consider the examples in Figure 19. Only three dots of constraints were provided, and the resulting matte is some interpolation from black to white, adapting to the image texture. The source of the problem is demonstrated by the synthetic example in the second row of Figure 19. In this example the input image consists of three regions, only two of which are constrained.

(a) Scribbles (b) Wang-Cohen (c) Our result



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

Fig. 17. An example with ambiguity between F and B.

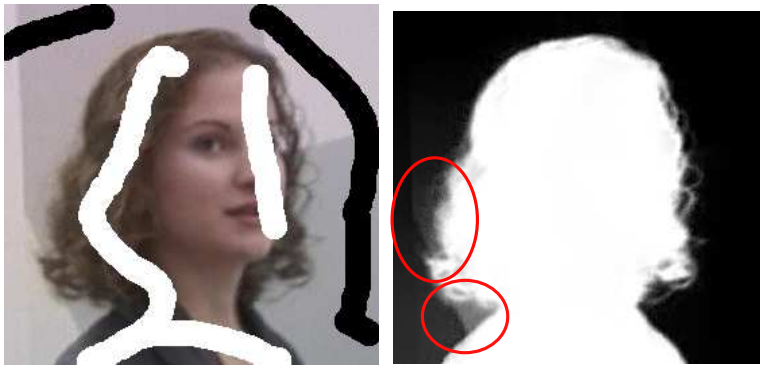


Fig. 18. Failure due to lack of a color model.

The system is then free to assign the middle unconstrained region any average non-opaque gray value. The core of this problem is that while our quadratic cost places strong assumptions on the foreground and background distributions, it imposes no restrictions on α. Thus, it searches for *continuous* solutions without taking into account that, for a mostly opaque foreground object,

the matte should be strictly zero or one over most of the image.

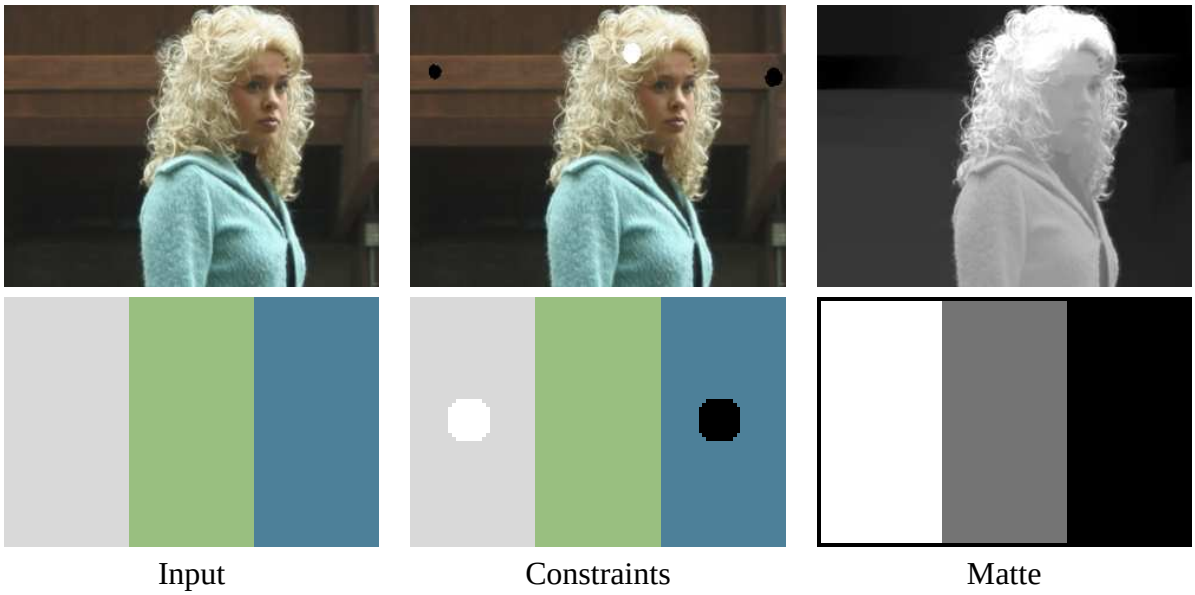


Fig. 19. Limitations in the lack of sufficient user input

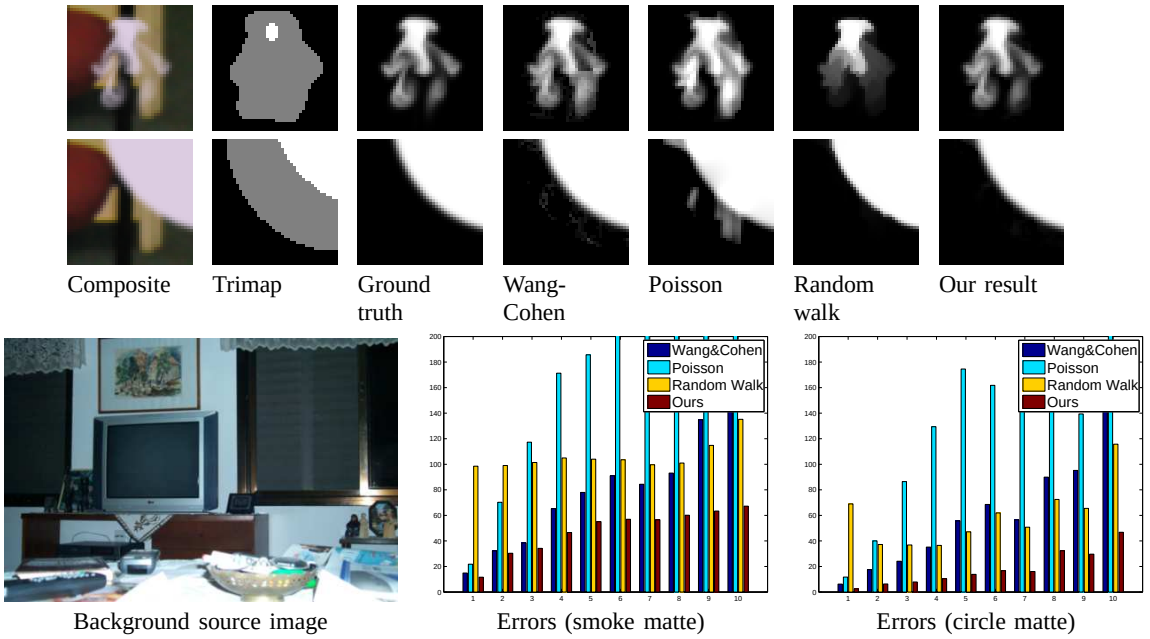


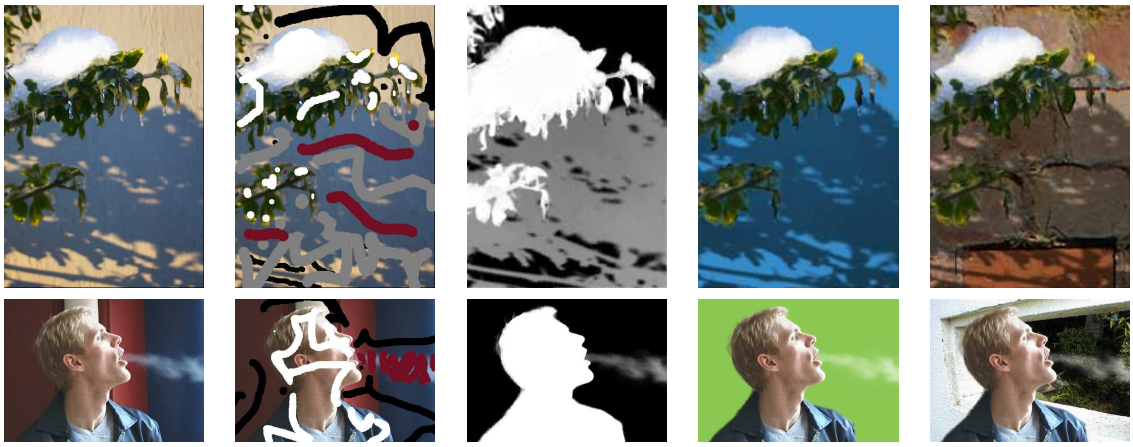
Fig. 20. A quantitative comparison using two ground truth mattes. The errors are plotted as a function of average gradient strength of the background, binned into 10 bins. To produce these results we used our own implementation of the respective methods, using the parameter values specified in the original papers.

B. Quantitative Comparisons

To obtain a 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 20. We used each subimage as a background and composited over it a uniform foreground image using two different alpha mattes: the first matte is computer simulated smoke, most of which is partially transparent; the other matte is a part of a disk, mostly opaque with a feathered boundary. The mattes are shown in Figure 20. Consequently, we obtained 4000 composite images, two of which are shown in Figure 20. 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 3×3 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 20. For each algorithm, we measured the summed absolute error between the extracted matte and the ground truth. Figure 20 plots 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). 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.

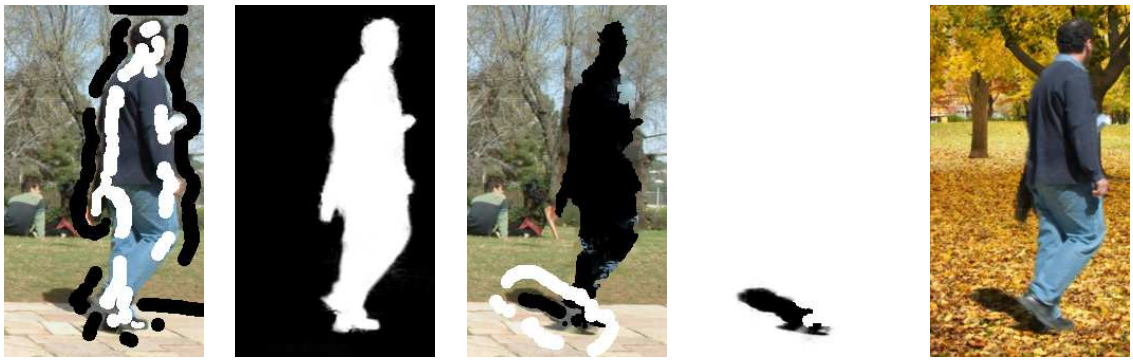
C. Shadow Matting

Figure 21 presents additional applications of our technique. In particular, the red marks specifying the foreground and background color, may be used to extract shadow and smoke. In the top row, the red scribbles on the shadow specify that the foreground color is black. In the bottom row, the red scribble on the smoke indicates the foreground color is white (in both cases the background color for the red scribbles was selected from neighboring, uncovered pixels). These sparse constraints on α were then propagated to achieve the final matte. Note that shadow matting can not be directly achieved with matting algorithms which initialize foreground colors using neighboring pixels, since no neighboring black pixels are present. Note also that the shadow area captures a significant amount of the image area and it's not clear how to specify



(a) (b) (c) (d) (e)

Fig. 21. Additional examples. Our technique applied for shadow and smoke extraction. (a) input image (b) scribbles (c) extracted mattes (d-e) composites.



(a) (b) (c) (d) (e)

Fig. 22. Shadow compositing with the shadow composition equation [7]. (a) marked input image (b) extracted α-matte (c) extracted background with marked shadow (d) β shadow matte. (e) compositing foreground and shadow with a novel background.

a good trimap in this case. The smoke example was processed also in [5], but in their case a background model was calculated using multiple frames.

An alternative approach for shadow extraction is to use the shadow composition equation proposed by [7]

$$I = \beta L + (1 - \beta)S$$

Where I is the input image, L the *lit image*, S the *shadow image* and β the *shadow density matte*. We consider the image in Figure 22. We first place black and white scribbles on the man and extract him from the background. We are then left with the background image in Figure 22(c) from which we would like to extract the shadow. This enables us to place black scribbles inside

the shadow area and white scribbles outside. Those scribbles are used for computing the shadow mask β in Figure 22(d). We can use the two mattes (Figure 22(b,d)) to paste both the man and his shadow over a novel background, as shown in Figure 22(e). The double compositing follows the following formula:

$$I_{new} = \alpha F + (1 - \alpha)(1 - s + s\beta B_{new})$$

where s is some scalar $0 < s < 1$ controlling the shadow strength. To the best of our knowledge, this is the first attempt to address shadow matting using an interactive interface.

IX. 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 requires 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 methods 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 function may be found efficiently by solving a sparse system 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. Furthermore, our results are competitive with those obtained by much more complicated, nonlinear, cost functions. However, compared to previous nonlinear 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.

Finally, the implementation of our matting algorithm and all examples presented in this paper are available for public usage at:

`http://people.csail.mit.edu/alevin/matting.tar.gz`

X. Acknowledgments

This work was supported by the Israel Science Foundation founded by the Israel Academy of Sciences and Humanities.

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] Yuri Boykov and Marie Pierre Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *Proc. ICCV*, 2001.

[4] Chakra Chennubhotla and Allan Jepson. Hierarchical eigensolver for transition matrices in spectral methods. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 273–280. MIT Press, Cambridge, MA, 2005.

[5] Y.Y. Chuang, A. Agarwala, B. Curless, D.H. Salesin, and R. Szeliski. Video matting of complex scenes. *ACM Trans. Graph.*, 21(3):243–248, 2002.

[6] Y.Y. Chuang, B. Curless, D.H. Salesin, and R. Szeliski. A Bayesian approach to digital matting. In *Proc. CVPR*, 2001.

[7] Y.Y. Chuang, D.B. Goldman, B. Curless, D.H. Salesin, and R. Szeliski. Shadow matting and compositing. *ACM Trans. Graph.*, 2003.

[8] Leo Grady, Thomas Schiwietz, Shmuel Aharon, and Rüdiger Westermann. Random walks for interactive alpha-matting. In *Proc. VIIP05*.

[9] Y. Guan, W. Chen, X. Liang, Z. Ding, and Q. Peng. Easy matting. In *Eurographics*, 2006.

[10] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 2004.

[11] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *ACM Trans. Graph.*, 23(3):303–308, 2004.

[12] Ido Omer and Michael Werman. Color lines: Image specific color representation. In *Proc. CVPR*, June 2004.

[13] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. ”grabcut”: interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004.

[14] M.A. Ruzon and C. Tomasi. Alpha estimation in natural images. In *Proc. CVPR*, 2000.

[15] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proc. CVPR*, pages 731–737, 1997.

[16] R. Szeliski. Locally adapted hierarchical basis preconditioning. *ACM Trans. Graph.*, 25(3):1135–1143, 2006.

[17] Jian Sun, Jiaya Jia, Chi-Keung Tang, and Heung-Yeung Shum. Poisson matting. *ACM Trans. Graph.*, 23(3):315–321, 2004.

[18] A. Torralba and W. T. Freeman. Properties and applications of shape recipes. In *Proc. CVPR*, June 2003.

[19] Jue Wang and Michael Cohen. An iterative optimization approach for unified image segmentation and matting. In *Proc. IEEE Intl. Conf. on Computer Vision*, 2005.

[20] A. Zomet and S. Peleg. Multi-sensor super resolution. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, 2002.