Spectral Matting

Anat Levin[1,2] Alex Rav-Acha[1,3] Dani Lischinski[1]

[1]School of CS & Eng
The Hebrew University of Jerusalem

[2]CSAIL
MIT

[3]CS & Applied Math
The Weizmann Institute of Science

Abstract

We present *spectral matting*: a new approach to natural image matting that automatically computes a set of fundamental fuzzy *matting components* from the smallest eigenvectors of a suitably defined Laplacian matrix. Thus, our approach extends spectral segmentation techniques, whose goal is to extract hard segments, to the extraction of soft matting components. These components may then be used as building blocks to easily construct semantically meaningful foreground mattes, either in an unsupervised fashion, or based on a small amount of user input.

I. Introduction

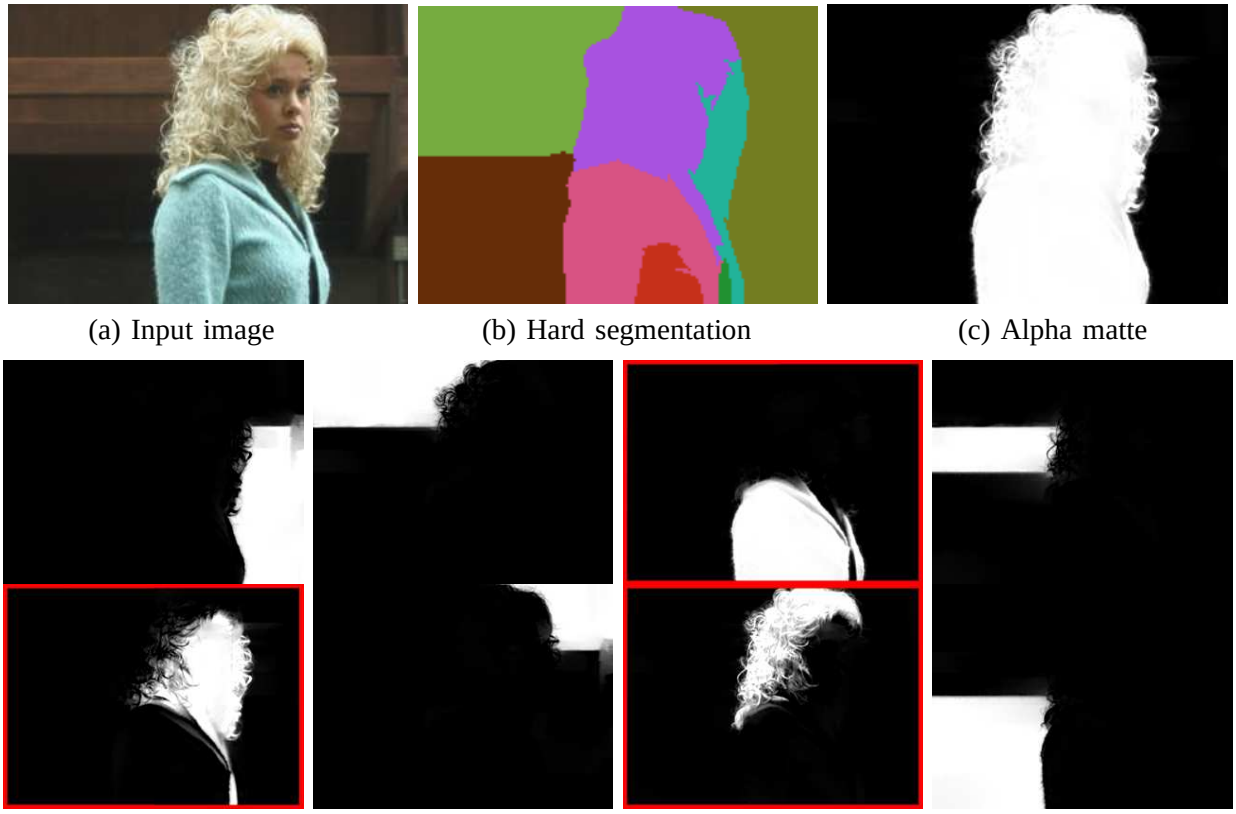
Digital matting is the process of extracting a foreground object from an image along with an opacity estimate for each pixel covered by the object. This operation enables compositing the extracted object over a novel background, and thus constitutes an invaluable tool in image editing, video production, and special effects in motion pictures.

In particular, the challenging case of *natural* image matting, which poses no restrictions on the background, has received much research attention. Recognizing that the problem is inherently under-constrained, all of the existing methods require the user to provide additional constraints in the form of a trimap [3], [21], [8] or a set of brush strokes [24], [14], [9]. Thus, the question of whether (or to what degree) is it possible to automate the matting process, is of considerable theoretical and practical interest.

In this paper we attempt to provide some new insights into this question. Our work is strongly influenced by spectral segmentation methods [19], [25], [17], [26]. These methods analyze the smallest eigenvectors of the image's graph Laplacian matrix in order to obtain an unsupervised decomposition of the image into a collection of hard segments. In this work, we extend this idea to unsupervised computation of a collection of soft *matting components*.

Spectral segmentation methods, such as [19], resort to computation of real-valued eigenvectors as an approximation necessary to transform an NP-complete optimization problem into a tractable one. In contrast, we are not seeking a disjoint image partitioning, but rather attempt to recover the fractional foreground coverage at each pixel. Specifically, we obtain our real-valued matting components via a linear transformation of the smallest eigenvectors of the *matting Laplacian* matrix, introduced by Levin *et al.* [14]. Once obtained, these matting components serve as building blocks for construction of complete foreground mattes.

This concept is illustrated in Figure 1. Given the input image in Figure 1a, one can produce an unsupervised disjoint hard partitioning of the image using, e.g., [26] (Figure 1b). In contrast, we compute



(a) Input image (b) Hard segmentation (c) Alpha matte

(d) Matting components computed by our method.

Fig. 1. Spectral segmentation and spectral matting

a set of overlapping, fractional, matting components, visualized in Figure 1d. Combining three of these components (framed in red) yields the foreground matte of the girl, shown in Figure 1c.

In summary, our main contribution is the introduction of the concept of fundamental matting components and a method for computing them in an unsupervised manner. We then proceed to describe an unsupervised matting algorithm. Of course, just like unsupervised segmentation, unsupervised matting is an ill-posed problem. Thus, we focus more on two extensions that use our fundamental matting components to construct a particular matte: (i) present the user with several matting alternatives to choose from; or (ii) let the user specify her intent by just a few mouse clicks.

A shorter version of this paper appeared in [15].

II. Matting components

Matting algorithms typically assume that each pixel I_i in an input image is a linear combination of a foreground color F_i and a background color B_i:

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

This is known as the *compositing equation*. In this work, we generalize the compositing equation by assuming that each pixel is a convex combination of K image layers $F^1, \ldots, F^K$:

$$I_i = \sum_{k=1}^{K} \alpha_i^k F_i^k. \tag{2}$$

The K vectors α^k are the *matting components* of the image, which specify the fractional contribution of each layer to the final color observed at each pixel. The matting components are non-negative and sum to one at every pixel. The intuitive motivation for having these components is that, similarly to the individual low-level fragments in an over-segmented image, they may be used to construct higher level, semantically meaningful foreground mattes, as demonstrated in Figure 1.

A desirable, although not required, property of the matting components is *sparsity*: each component should be either completely opaque or completely transparent over as many image pixels as possible. This means that areas of transition between the different layers are limited to a small number of pixels, and each pixel is influenced by a small number of layers.

In this paper, we explore the relationship between the matting components and the eigenvectors of the matting Laplacian matrix [14]. Specifically, we show that under certain assumptions the matting components are spanned by the smallest eigenvectors of the matting Laplacian. We then propose a method for computing the matting components by finding an appropriate linear transformation and applying it to these eigenvectors.

III. Spectral Analysis

We start by briefly reviewing the basic theory of spectral segmentation methods [5], [6], [7], [23], [11], [1], [10], [19], [25], [17], [26]. These methods typically associate with the image an $N \times N$ affinity matrix A, such as $A(i,j) = e^{-d_{ij}/\sigma^2}$, where d_{ij} is some measure of the distance between the pixels (such as color difference and geometric distance). One can then define the Laplacian matrix $L = D - A$, where D is the diagonal matrix $D(i,i) = \sum_j A(i,j)$. L is a symmetric positive semidefinite matrix, whose eigenvectors

capture much of the image structure.[1]

Consider the ideal case where the affinity matrix A captures exactly the fact that an image is composed from several distinct clusters, or *connected components*. That is, a subset C of the image pixels is a connected component of the image if $A(i,j) = 0$ for every i, j such that $i \in C,\ j \notin C$, and there is no subset of C which satisfies this property. Let m^C denote the indicator vector of the component C,

$$m_i^C = \begin{cases} 1 & i \in C \\ 0 & i \notin C \end{cases},$$

then m^C is a 0-eigenvector of L (i.e., an eigenvector with eigenvalue 0).

Now suppose that the image consists of K connected components $C_1, \ldots, C_K$ such that $\{1, \ldots, N\} = \bigcup_{k=1}^{K} C_k$, where C_k are disjoint subsets of pixels. In this case the indicator vectors $m^{C_1}, \ldots, m^{C_K}$ are all independent, orthogonal 0-eigenvectors of L. However, computing the eigenvectors of L yields these indicator vectors only up to rotation. This is the case since for any $K \times K$ rotation matrix R the vectors $[m^{C_1}, \ldots, m^{C_K}]R$ are also a basis for the nullspace of L.

In real images, the affinity matrix A is rarely able to perfectly separate between the different pixel clusters. Therefore, the Laplacian L usually does not have multiple 0-eigenvectors. However, it has been observed that the smallest eigenvectors of L tend to be nearly constant within coherent image components. Extracting the different components from the smallest eigenvectors is known as *spectral rounding* and has attracted much attention [17], [26], [22], [27], [13]. The simplest approach [17] is to cluster the image pixels using the k-means algorithm, and use perturbation analysis to bound the error of this algorithm as a function of the connectivity within and between clusters. Other more recent methods [26], [27], which inspired the approach taken in this work, explicitly search for a rotation matrix that brings the eigenvectors as close as possible to binary indicator vectors.

A. Spectral Analysis with the Matting Laplacian

Our goal in this work is to derive an analogy between hard segmentation and matting and to show that fuzzy matting components may be extracted from the smallest eigenvectors of the matting Laplacian, similarly to the extraction of hard clusters described earlier.

[1] In fact, most spectral segmentation papers consider normalized affinity matrices such as $D^{-1}L$ or $D^{-1/2}LD^{-1/2}$. However, in this work we focus on L itself as it is not clear how to justify the normalization in the case of the matting Laplacian. The problem is that the off-diagonal elements of the matting Laplacian can be both negative and positive, and thus the matting cost cannot be expressed as a sum of positive pairwise terms.

1) The Matting Laplacian: The matting Laplacian was introduced by Levin *et al.* [14] in order to evaluate the quality of a matte without explicitly estimating the foreground and background colors in eq. (1). They show that if the colors of the background and the foreground within a local image window w form two different lines in RGB space, then the α values within w may be expressed as a linear combination of the color channels:

$$\forall i \in w \qquad \alpha_i = a^R I_i^R + a^G I_i^G + a^B I_i^B + b \tag{3}$$

Thus, the matte extraction problem becomes one of finding the alpha matte that minimizes the deviation from the linear model (3) over all image windows w_q:

$$J(\alpha, a, b) = \sum_{q \in I} \sum_{i \in w_q} \left(\alpha_i - a_q^R I_i^R - a_q^G I_i^G - a_q^B I_i^B - b_q\right)^2 + \varepsilon \|a_q\|^2 \tag{4}$$

where $\varepsilon \|a_q\|^2$ is a regularization term on a. The linear model coefficients a, b may be eliminated from equation (4), yielding a *quadratic cost in α alone*,

$$J(\alpha) = \alpha^T L \alpha. \tag{5}$$

This cost has a trivial minimum, which is a constant α vector, and thus in the user assisted framework described in [14], $J(\alpha)$ is minimized subject to user constraints.

In eq. (5), L is the *matting Laplacian*, a sparse symmetric positive semidefinite $N \times N$ matrix whose entries are a function of the input image in local windows, depending neither on the unknown foreground and background colors, nor on the linear model coefficients. L is defined as a sum of matrices $L = \sum_q A_q$, each of which contains the affinities among pixels inside a local window w_q:

$$A_q(i,j) = \begin{cases} \delta_{ij} - \frac{1}{|w_q|}\left(1 + (I_i - \mu_q)^T (\Sigma_q + \frac{\varepsilon}{|w_q|} I_{3\times 3})^{-1} (I_j - \mu_q)\right) & (i,j) \in w_q \\ 0 & \text{otherwise} \end{cases}, \tag{6}$$

Here δ_{ij} is the Kronecker delta, μ_q is the 3×1 mean color vector in the window w_q around pixel q, Σ_q is a 3×3 covariance matrix in the same window, $|w_q|$ is the number of pixels in the window, and $I_{3\times 3}$ is the 3×3 identity matrix.

Note that the matting Laplacian is fundamentally different from a "standard" graph Laplacian matrix. Standard Laplacian matrices are constructed using non-negative affinities $A(i,j)$, and hence all off-diagonal elements of L are non-positive. On the other hand, according to eq. (6), the off-diagonal elements of the matting Laplacian may have arbitrary signs. Thus, although we follow the terminology of Levin

et al. [14], and refer to the matrix L as the matting Laplacian, it should be understood that this is an extension of the standard term, as most important properties of graph Laplacians rely on the non-negative affinity assumption.

Fortunately, as we show below, some of the useful properties of standard graph Laplacian matrices apply to the matting Laplacian as well. The first such property is that both types of Laplacians are positive semidefinite matrices. For the standard graph Laplacian, this property is trivially implied from the fact that the pairwise affinities $A(i,j)$ are non-negative; for every N-dimensional vector x, it holds that $x^T L x = \sum_{i,j} A(i,j)(x_i - x_j)^2 \geq 0$. The matting Laplacian is also positive semidefinite, but instead of positive pairwise terms, it can be factored as a sum of positive terms consisting of the affinities in 3×3 windows; by construction of the matting Laplacian (see [14] for the exact derivation), for every N-dimensional vector x, it holds that

$$x^T A_q x = min_{a,b} \sum_{i \in w_q} \left(x_i - a_q^R I_i^R - a_q^G I_i^G - a_q^B I_i^B - b_q \right)^2 + \varepsilon \|a_q\|^2 \geq 0,$$

which implies $x^T L x = \sum_q x^T A_q x \geq 0$.

Another useful property of the matting Laplacian (6) is that its smallest eigenvectors appear to capture information about the fuzzy cluster assignments of pixels in the image, even before any user-specified constraints are taken into account. This observation was already made by Levin *et al.* [14]; however, they made no use of the eigenvectors beyond presenting them to the user as guides for scribble placement. In this work, we show that the smallest eigenvectors of the matting Laplacian span the individual matting components of the image.

2) The Matting Laplacian's Nullspace: To gain some understanding, we begin by studying the ideal case. To justify the usage of spectral analysis to estimate matting components, our goal is to show that under reasonable conditions, the actual matting components belong to the nullspace of the matting Laplacian. We say that a matting component α^k is *active* in a local image window w if there exists a pixel $i \in w$ for which $\alpha_i^k > 0$. The following claim states the conditions on the local color distribution in each layer, under which $L\alpha^k = \mathbf{0}$. The severity of the conditions is related to the number of active layers in a local window. The least restricted case is when only one layer is active, in which the local color distribution can be arbitrary complex. The most restricted case is when a window contains three active layers (as in the case of a T-junction), and for such windows the theorem holds when each layer color is locally uniform.

Claim 1: Let $\alpha^1, \ldots, \alpha^K$ be the actual decomposition of the image I into k matting components. The

vectors $\alpha^1,\ldots,\alpha^K$ lie in the nullspace of the matting Laplacian L (given by eq. 6 with $\varepsilon = 0$) if every local image window w satisfies one of the following conditions:

1) A single component α^k is active within w.
2) Two components $\alpha^{k_1},\alpha^{k_2}$ are active within w and the colors of the corresponding layers F^{k_1},F^{k_2} within w lie on two different lines in RGB space.
3) Three components $\alpha^{k_1},\alpha^{k_2}$ and α^{k_3} are active within w, each layer F^{k_1},F^{k_2},F^{k_3} has a constant color within w, and the three colors are linearly independent.

Proof: The matting cost (5) measures the deviation between a matte and a linear function of the color channels, over all local windows (eq. 4). Thus, in order to show that a matte component α^k satisfies $L\,\alpha^k = \mathbf{0}$ it suffices to show that for every local window w, there exist a^R,a^G,a^B,b such that: $\alpha_i^k = a^R I_i^R + a^G I_i^G + a^B I_i^B + b, \;\; \forall i \in w$. Below we show this for each of the three window types.

Case 1: Since the matting components sum to one at every image pixel, the single active component α^k must equal 1 within w. Thus, it is easily expressed as a linear function of the image by setting $a^R = a^G = a^B = 0$ and $b = 1$.

Case 2: This case is equivalent to theorem 2 in [14].

Case 3: Assume wlog that $k_1 = 1$, $k_2 = 2$, $k_3 = 3$. Let $\mathbf{F} = [F^1,F^2,F^3]$ be a 3×3 matrix of the uniform layer colors, and for $i \in w$ let $\alpha_\mathbf{i} = [\alpha_i^1,\alpha_i^2,\alpha_i^3]^T$ be a 3×1 vector of components values. We note that $I_i = \mathbf{F}\alpha_\mathbf{i}$. Since the 3 layer colors are linearly independent, $\mathbf{F}$ is invertible and $\alpha_\mathbf{i} = \mathbf{F}^{-1} I_i$, which implies that α^k are linear functions of the image. □

As in the case of standard Laplacians, when the smallest eigenvectors of the matting Laplacian are computed, the result may be any linear combination of the different matting components, and recovering the individual components is equivalent to linearly transforming the eigenvectors. It should be noted that unlike hard segments, the matting components are not binary vectors and thus are not necessarily orthogonal. Hence, while the eigenvectors are orthogonal, the transformation from eigenvectors to matting components might be a general linear transformation and not a simple rotation.

To summarize, the main conclusion of the above discussion is that whenever the matting components of an image satisfy the conditions of claim 1, they may be expressed as a linear combination of the 0-eigenvectors of L.

In most real images, the assumptions of claim 1 don't hold exactly, and thus the matting Laplacian might not have multiple 0-eigenvectors. Yet if the layers are sufficiently distinct, they are generally captured by the smallest eigenvectors of L. For example, Figure 2 shows the smallest eigenvectors for a real image, all exhibiting the fuzzy layer boundaries. We have empirically observed that the matting components of

Fig. 2. The smallest eigenvectors of the matting Laplacian for the image in Figure 1a. Linear combinations of these eigenvectors produced the matting components shown in Figure 1d.

real images are usually spanned quite well by the smallest eigenvectors of the matting Laplacian. Indeed, the components shown in Figure 1d were obtained as linear combinations of the smallest eigenvectors.

B. From Eigenvectors to Matting Components

As explained above, recovering the matting components of the image is equivalent to finding a linear transformation of the eigenvectors. Recall that the matting components should sum to 1 at each image pixel, and they should be near 0 or 1 for most image pixels, since the majority of image pixels are usually opaque. Thus, we are looking for a linear transformation of the eigenvectors that would yield a set of nearly binary vectors. More formally, let $E = [e^1, .., e^K]$ be the $N \times K$ matrix of eigenvectors. Our goal is then to find a set of K linear combination vectors y^k that minimize

$$\sum_{i,k} |\alpha_i^k|^\gamma + |1-\alpha_i^k|^\gamma, \text{ where } \alpha^k = Ey^k \qquad (7)$$
$$\text{subject to } \sum_k \alpha_i^k = 1$$

If $0 < \gamma < 1$ is used (in our implementation $\gamma = 0.9$), then $|\alpha_i^k|^\gamma + |1-\alpha_i^k|^\gamma$ is a robust score measuring the sparsity of a matting component (plotted in Figure 3). Without the requirement $\alpha^k = Ey^k$ the sparsity term would be minimized by binary vectors, but as the vectors α^k are restricted to linear combinations of the eigenvectors they must maintain the fuzzy layer boundaries. Although we do not explicitly constrain the α values to be between 0 and 1, in practice the resulting values tend to lie in this range due to the sparsity penalty.

The above cost is of course a non-convex one and we optimize it iteratively using Newton's method [4],

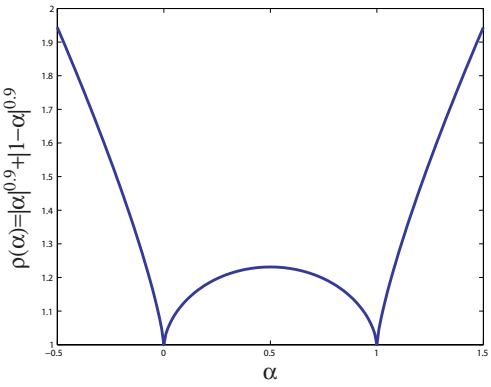


Fig. 3. Sparsity score: $\rho(\alpha) = |\alpha|^{0.9} + |1-\alpha|^{0.9}$ favoring binary α values

which amounts to a sequence of second order approximations. Given a guess for the values of α_i^k we define u_i^k as the second derivative of $|\alpha_i^k|^\gamma$, $\quad u_i^k \propto |\alpha_i^k|^{(\gamma-2)}$, and similarly $v_i^k \propto |1-\alpha_i^k|^{(\gamma-2)}$. The second order approximation to eq. (7) reads as minimizing

$$\begin{aligned} &\sum_{i,k} u_i^k |\alpha_i^k|^2 + v_i^k |1-\alpha_i^k|^2, \text{ where } \alpha^k = Ey^k \\ &\text{subject to } \sum_k \alpha_i^k = 1 \end{aligned} \tag{8}$$

As this problem has now become one of quadratic optimization under linear constraints, the optimal solution can be computed in closed form by inverting a $K^2 \times K^2$ matrix. For that we define $U^k = diag(u^k)$ and $V^k = diag(v^k)$. We also define $W^k = E^T(U^k + V^k)E$ and $\mathbf{1}$ as a vector of ones. It may be shown that the K^2 elements of $y^1, \ldots, y^K$ are the solution for:

$$\begin{bmatrix} \mathrm{Id} & \mathrm{Id} & \mathrm{Id} & \cdots & \mathrm{Id} \\ 0 & W^2+W^1 & W^1 & \cdots & W^K \\ 0 & W^1 & W^3+W^1 & \cdots & W^K \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & W^1 & W^1 & \cdots & W^K+W^1 \end{bmatrix} y = \begin{bmatrix} E\mathbf{1} \\ Ev^2+Eu^1 \\ Ev^3+Eu^1 \\ \vdots \\ Ev^K+Eu^1 \end{bmatrix}, \tag{9}$$

where Id is the $K \times K$ identity matrix. Given an initial guess, our algorithm iteratively solves a sequence of second order optimization problems of the form of eq. (8) and uses the solution of each iteration to reweight the following one. We note that the weights u_i^k (and v_i^k) are higher when the current guess for α_i^k is close to 0 (or 1) and are lower when the guess is farther away from these values. Thus, the effect

of the reweighting step is to pull toward 0 (or 1) those α entries for which the current guess is already close, and to loosen the relation for pixels for which the guess is far anyway. For example, if the current guess for α_i^k is close to 0, then $u_i^k \gg v_i^k$ and thus the term $u_i^k|\alpha_i^k|^2 + v_i^k|1-\alpha_i^k|^2$ pulls the alpha component at this pixel toward 0 much stronger than toward 1.

Since the cost (7) is not convex, the result of the Newton process strongly depends on the initialization. One useful way to initialize the process is to apply a k-means algorithm on the smallest eigenvectors of the matting Laplacian and project the indicator vectors of the resulting clusters onto the span of the eigenvectors E:

$$\alpha^k = EE^T m^{C^k}. \tag{10}$$

The effect of this projection step is to find a linear combination of eigenvectors that minimizes the squared distance to m^{C^k}. We note that the resulting matting components sum to one, since

$$\sum_k \alpha^k = EE^T(\sum_k m^{C^k}) = EE^T\mathbf{1} = \mathbf{1}. \tag{11}$$

The last equality follows from the fact that the constant vector belongs to the nullspace of the matting Laplacian. As a result, projecting the hard clusters provides a legal solution for eq. (7). We also note that despite the fact that m^{C^k} are binary vectors, the projection typically features fuzzy matte boundaries. This is due to the fact that, as illustrated in Figure 2, the smallest eigenvectors all maintain the fuzzy layer structure, and thus simply do not suffice to span the hard segmentation boundaries.

In practice, we typically use a larger number of eigenvectors than the number of matting components to be recovered. Using more eigenvectors makes it possible to obtain sparser components. The reason is that more basis elements span a richer set of vectors (in the extreme case, if all N eigenvectors are used, any binary vector can be generated). A number of examples demonstrating the extraction of soft matting components are given in Figures 1 and 4.

IV. Grouping Components

So far we have shown how matting components may be extracted from the matting Laplacian. However, usually the matting components are not a goal in their own, as one is ultimately interested in recovering a complete matte for some foreground object. Fortunately, all that is needed to obtain a complete matte is to specify which of the components belong to the foreground. Let b denote a K-dimensional binary vector indicating the foreground components (i.e. $b^k = 1$ iff α^k is a foreground component). The complete

Fig. 4. A number of test images and the matting components extracted from them using our method. A random color is assigned to each component for visualization purposes.

foreground matte is then obtained by simply adding the foreground components together:

$$\alpha = \sum_k b^k \alpha^k \tag{12}$$

For example, the matte in Figure 1c was obtained by adding the components highlighted in red in Figure 1d.

For the applications discussed below, one would like to compare multiple grouping hypotheses. If the smallest eigenvalues aren't exactly zero (which is the case for most real images) we can also measure the quality of the resulting α-matte as $\alpha^T L \alpha$, where L is the matting Laplacian (eq 6). When a large number of hypotheses is to be tested, multiplying each hypothesis by L might be too expensive. However, if each hypothesis is just a sum of matting components we can pre-compute the correlations between the matting components via L and store them in a $K \times K$ matrix Φ, where

$$\Phi(k,l) = {\alpha^k}^T L \, \alpha^l. \tag{13}$$

The matte cost can then be computed as

$$J(\alpha) = b^T \Phi \, b, \tag{14}$$

where b is a K dimensional binary vector indicating the selected components. Thus, if Φ has been pre-computed, $J(\alpha)$ can be evaluated in $O(K^2)$ operations instead of $O(N)$ operations.

Fig. 5. Unsupervised matting results for a few images. The hypotheses are ordered according to their score.

A. Unsupervised Matting

Given an image and a set of matting components we would like to split the components into foreground and background groups and pull out a foreground object. If the grouping criterion takes into account only low level cues, then we just search for a grouping with the best matting cost, as defined by eq. (14). However, the matting cost is usually biased toward mattes which assign non constant values only to a small subset of the image pixels (in the extreme case, the best matte is a constant one). The spectral segmentation literature suggest several criteria which overcome this bias. One approach is to search for quotient cuts (e.g. normalized cuts [19]) which score a cut as the ratio between the cost of the cut and the size of the resulting clusters. A second approach is to look for balanced cuts [13] where the size of

each cluster is constrained to be above a certain percent of the image size. In this work, we follow this latter approach and rule out trivial solutions by considering only groupings which assign at least 30% of the pixels to the foreground and at least 30% of the pixels to the background. When the number K of matting components is small we can enumerate all 2^K hypotheses and select the one with the best score using eq. (14).

Figure 5 shows some results produced by the unsupervised matting approach described above. In two of these examples the hypothesis with the highest score indeed corresponds to the "correct" foreground matte, while in one example (bottom row) the "correct" matte was ranked fourth. Note that in all of these examples the other high-ranked hypotheses are quite sensible as well, considering that our approach does not attempt to perform any high-level image understanding. Of course, it isn't hard to find examples where unsupervised matting fails, and the last two examples in figure 5 illustrate such failures. In general, whenever the foreground or background objects consist of several visually distinct components, the assignment with the minimal matting cost might not correspond to our visual perception. In fact, it is well known within the image segmentation community that while unsupervised bottom-up cues can efficiently group coherent regions in an image, the general image segmentation problem is inherently ambiguous, and requires additional information. In practice, such as in the case of hard image segmentation, the foreground/background assignment may be guided by several additional cues, such as top-down models [2], color statistics [18], or motion and focus cues. Thus, we believe that the main practical use of matting components is in the supervised setting, and focus on user-guided matting in the remainder of this paper.

B. User-Guided Matting

We now consider an interactive setting, where the user guides the matting process toward the extraction of the desired foreground matte. In such a setting the foreground/background assignment of some of the components is determined by the user, thereby reducing the number of legal hypotheses to be tested. Given very minimal foreground and background constraints, it is usually possible to rule out trivial solutions, so there is no need to explicitly keep the size of each group above a certain threshold (as in the unsupervised case). In this case we can speed up the search for an optimal foreground/background assignment of components using a graph min-cut formulation.

1) Optimizing assignments in the components graph: To efficiently search for the optimal foreground/background assignment we approximate the matting cost (eq. 14) as a sum of pairwise terms. This enables us to approximate the search for the optimal foreground/background assignment as a min-cut

problem in a graph whose nodes are the matting components, and whose edge weights represent matting penalty. Since the function we are trying to minimize is not sub-modular in general, we rewrite eq. (14) as a sum of non-negative local and pairwise terms $J(\alpha) = E(b)$, where b is the binary vector representing the components in α. Alternatively, one could use algorithms for minimizing non-submodular functions such as the QPBO method. (see [12] for a review on such methods).

We define the energy $E(b)$ as

$$E(b) = \sum_k E_k(b^k) + \sum_{k,l} E_{k,l}(b^k - b^l)^2 \tag{15}$$

where $E_k(0) = \infty$ if the k-th component is constrained to belong to the foreground, $E_k(1) = \infty$ if the component is constrained as background, and 0 otherwise. To define the pairwise term we note that $\phi^{k,k} = \sum_{l \neq k} \phi^{k,l}$ (this follows from the fact that $L(\sum \alpha_k) = L\mathbf{1} = \mathbf{0}$, and thus $\Phi\mathbf{1} = \mathbf{0}$) and as a result

$$b^T \Phi b = \sum_{k,l} -\phi^{k,l}(b^k - b^l)^2. \tag{16}$$

Therefore, if we define $E_{k,l} = -\phi^{k,l}$ we obtain that

$$\sum_{k,l} E_{k,l}(b^k - b^l)^2 = J(\alpha). \tag{17}$$

However, in order to search for a min-cut efficiently, we use a positive approximation and define $E_{k,l} = \max(0, -\phi^{k,l})$. To justify this approximation we can prove the approximation is exact in all local image windows for which no more than two components are active. When good components are extracted from an image, the majority of image windows will be associated with no more than two components (that is, no more than two components will be active). Indeed, we have empirically observed that for most matting component pairs $\phi^{k,l} < 0$.

Claim 2: The correlation $\phi^{k,l}$ between components α_k, α_l will be negative if every local image window w satisfies one of the following conditions:

(c1) $\alpha_i^k = 0 \ \forall i \in w$

(c2) $\alpha_i^l = 0 \ \forall i \in w$

(c3) $\alpha_i^k = 1 - \alpha_i^l \ \forall i \in w$

Proof: Following eq. (6), we rewrite the correlation between components via the matting Laplacian as a sum of correlations over local windows:

$$\phi^{k,l} = \alpha^{k^T} L \alpha^l = \sum_{q \in I} \alpha^{k^T} L_q \alpha^l, \tag{18}$$

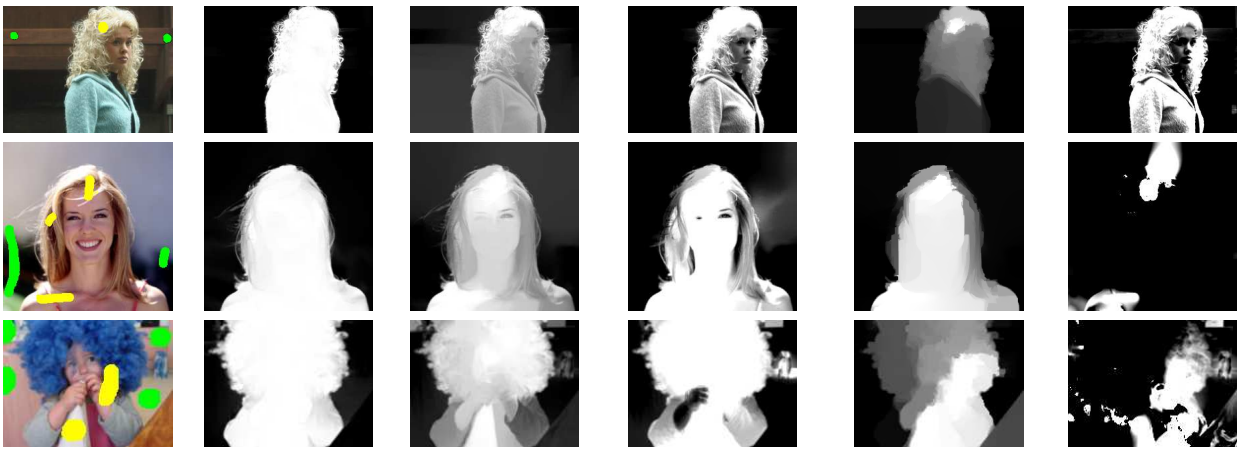


Fig. 6. A comparison of mattes produced by different matting methods from minimal user input (yellow scribbles indicate foreground, while green scribbles indicate background).

where L_q is an $N \times N$ matrix, if $(i,j) \in w_q$ $L_q(i,j)$ is defined using eq. (6), and $L_q(i,j) = 0$ otherwise. We will show that under the above conditions, ${\alpha^k}^T L_q \alpha^l < 0$ for every image window w_q. This will follow immediately if $\alpha_i^k = 0$ or $\alpha_i^l = 0 \; \forall i \in w$. For the third case, we note that

$$ {\alpha^k}^T L_q \alpha^l =^* {\alpha^k}^T L_q (\mathbf{1} - \alpha^k) =^{**} -{\alpha^k}^T L_q \alpha^k \leq^{***} 0, \tag{19} $$

Where * follows from the condition (c3), ** follows from the fact that the matting cost of the constant vector is 0, and *** follows from the fact that each of the L_q matrices is positive semidefinite. □

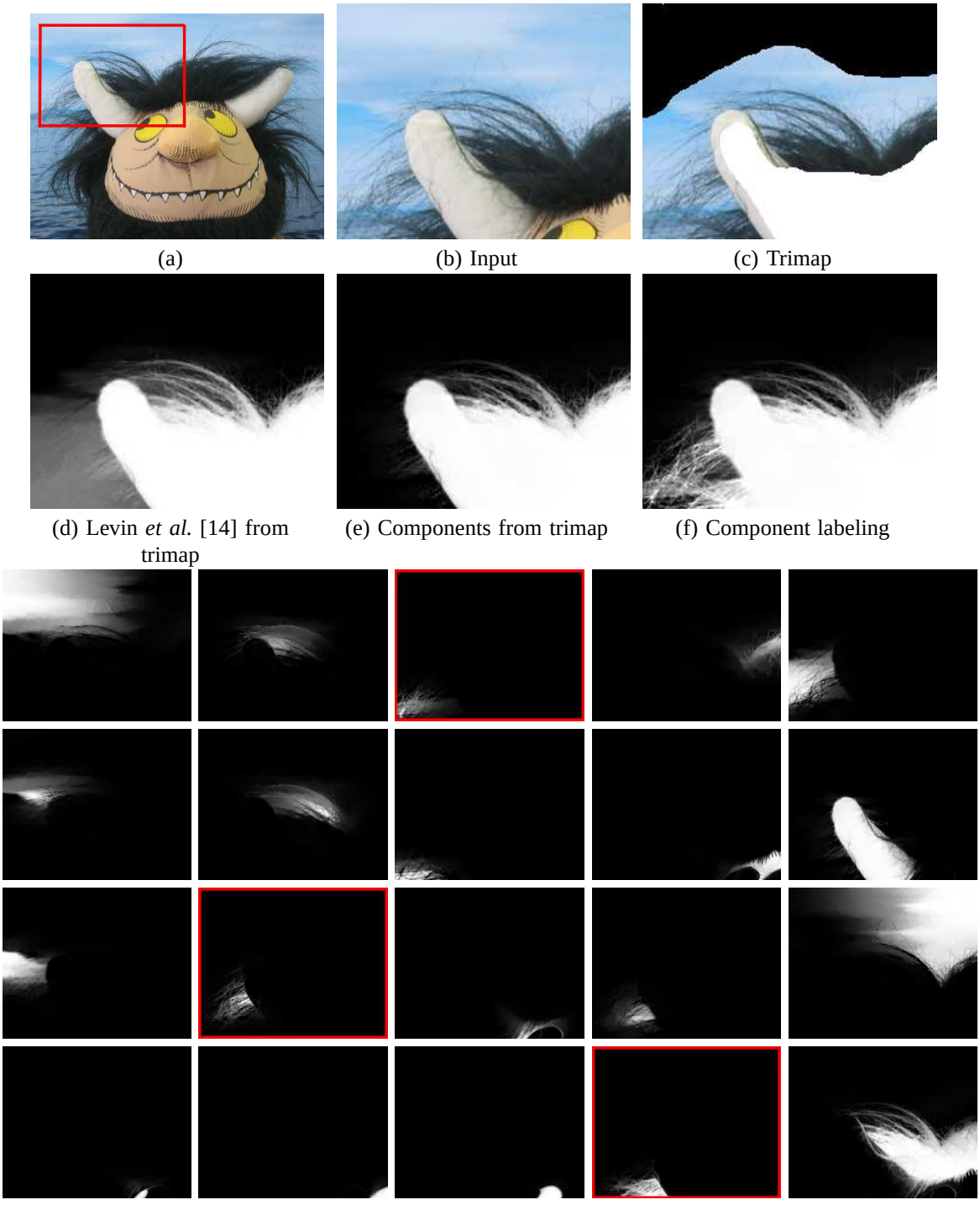
Using the above graph formulation, finding the optimal foreground/background assignment does not involve an exponential search and is found efficiently in time polynomial in the number of components, as a graph min-cut. As a result, if the matting components are pre-computed, the optimal matte may be computed very rapidly, enabling interactive responses to user input. The computational challenges of our algorithm are equivalent to those of conventional spectral segmentation techniques. Specifically, it takes our unoptimized matlab implementation a couple of minutes to compute the matting components for the images in Figure 6. However, this pre-processing step can be done offline, and once the matting components are available, it only takes an additional few seconds to construct a matte given the user's constraints.

2) Matte extraction using user scribbles: Figure 6 presents a few examples where a foreground matte was extracted from an image based on a small number of foreground (white) and background (black) markings provided by the user. The second column shows the resulting matte extracted by the approach described above (the scribbles are used to reduce the space of splitting hypotheses: a component is

Fig. 7. The middle region is not constrained, and the method of Levin *et al.* assigns it an average non-opaque value.

constrained to belong to the foreground whenever its area contains a white scribble). The remaining columns show the mattes generated from the same input by a number of previous methods [14], [24], [8], [21]. None of these previous approaches is able to recover a reasonable matte from such minimal user input. In particular, although our approach uses the same matting Laplacian as [14], our results are very different from those obtained by directly minimizing the quadratic matting cost (5) subject to user-specified constraints. The main drawback of such direct optimization is that whenever an image contains distinct connected components without any constraints inside them, a quadratic cost such as (5) tends to assign them some average non-opaque values, as demonstrated by the simple example in Figure 7. The core of this problem is that the quadratic cost of [14] places strong assumptions on the foreground and background distributions, but 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 0 or 1 over most of the image.

3) Matte extraction by component labeling: Once the matting components of an image have been computed, placing hard constraints by a set of scribbles or a trimap is not the only way for the user to specify her intent. The matting components suggest a new, more direct user interaction mode which wasn't possible until now: in this mode the user is presented with the precomputed matting components and may simply label some of them as background or foreground. The labeled components then become constrained accordingly in the min-cut problem. The advantage of such an interface is illustrated in Figure 8, where the large fuzzy hair areas do not lend themselves to placement of hard constraints. Thus, the best trimap we could practically expect leaves such areas unconstrained (Figure 8c). The least squares matte of [14] populates these areas with average gray values (Figure 8d). In contrast, by searching for the cheapest assignment of matting components consistent with the trimap, we obtain the matte in Figure 8e. In this case no over-smoothing is observed, but some of the fuzzy hair was not selected to belong to the foreground. However, if the user is allowed to directly select three additional components (highlighted in red in Figure 8g) as foreground, we obtain the matte in Figure 8f.



(a) (b) Input (c) Trimap

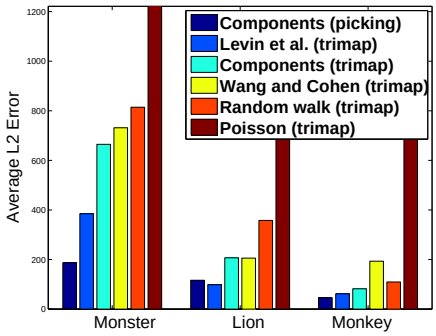
(d) Levin *et al.* [14] from trimap (e) Components from trimap (f) Component labeling

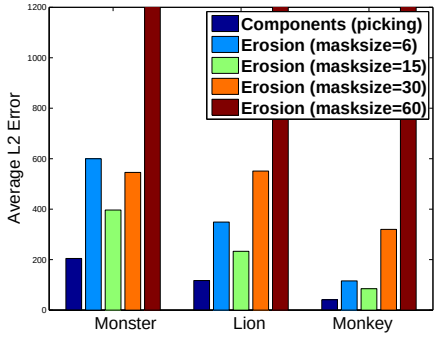
(g) Matting components

Fig. 8. Benefits of direct component labeling.

V. Quantitative evaluation

To quantitatively evaluate our approach and compare it with previous methods we captured ground truth data. Three different dolls were photographed in front of a computer monitor displaying seven





(a) A few test images (b) A comparison with other matting methods (c) Spectral matting vs. hard segmentation

Fig. 9. Quantitative evaluation

different background images (Figure 9a). A ground truth matte was then extracted for each doll using a least squares framework [20]. Each image was downsampled to 560×820 pixels, and the tests described below were performed on (overlapping) 200×200 windows cropped from these images. For our approach, 60 matting components were extracted using the 70 smallest eigenvectors of each cropped window. The running time of our unoptimized matlab implementation (on a 3.2GHz CPU) was a few minutes for each 200×200 window.

To design a comparison between matte extraction using matting components and previous matting algorithms we need to address the two non compatible interfaces, and it is not clear how to measure the amount of user effort involved in each case. While previous approaches were designed to work with hard constraints (scribbles or trimap) our new approach enables a new interaction mode by component selection. Therefore, in our experiments we attempted to determine how well can each approach do, given the best possible user input. Thus, we first used the ground truth matte to generate an "ideal" trimap. The unknown region in this trimap was constructed by taking all pixels whose ground truth matte values are between 0.05 and 0.95, and dilating the resulting region by 4 pixels. The resulting trimap was used as input for four previous matting algorithms: Levin *et al.* [14], Wang and Cohen [24], random walk matting [8], and Poisson matting [21]. We also ran our method twice in each experiment: (i) using the same trimap to provide a partial labeling of the matting components, followed by a min-cut computation, as described in section IV-B; and (ii) using the ground truth matte to select the subset of matting components that minimizes the distance of the resulting matte from the ground truth, thus simulating the ideal user input via the direct component picking interface. The SSD errors between the mattes produced by the different methods and the ground truth matte (averaged over the different backgrounds and the different windows) are plotted in Figure 9b. It is apparent that given a sufficiently precise trimap, our method offers no

real advantage (when given the same trimap as input) over the least-squares matting of Levin *et al.*, which produced the most numerically accurate mattes. However, when simulating the best labeling of components, our approach produced the most accurate mattes, on average.

While our experiment compares the quality of mattes produced from an ideal input, a more interesting comparison might be to measure the amount of user time required for extracting a satisfactory matte with each approach. Ideally, we would also like to measure whether (or to what degree) a component picking interface is more intuitive than a scribble based interface. Such a comparison involves a non trivial user study, and is left for future work.

Given the strong analogy between spectral matting and hard spectral segmentation, we would like to gain some intuition about the possible advantage of using matting components versus standard hard segmentation components (also known as super-pixels). The answer, of course, depends on the application. If the final output is a hard segmentation, matting components probably do not offer an advantage over standard hard components. On the other hand, when the goal is a fuzzy matte it is better to explicitly construct matting components, as we do, rather than first compute a hard segmentation and then feather the boundaries (as in [16], [18], for example). To show this, we compare the two approaches. We first extract matting components from each of the test images and select the subset of matting components, which will minimize the distance from the ground truth matte. The second approach is to select a subset of hard components (we used the available implementation of Yu and Shi [26]) that best approximates the ground truth. We then apply morphological operations (we have experimented with several constant radius erosion windows) on the resulting hard mask, create a trimap and run the matting algorithm of [14]. However, since the optimal radius of the erosion window strongly depends on the local image structure and varies over the image, it is impossible to obtain an ideal trimap with a constant radius window. This problem is illustrated visually in Figure 10.

Figure 9c shows the SSD errors (averaged over the different backgrounds and the different windows) of the two approaches, which indicate that optimally picking the matting components indeed results in more accurate mattes than those obtained by feathering a hard segmentation.

VI. Discussion

In this work we have derived an analogy between standard (hard) spectral image segmentation and image matting, and have shown how fundamental matting components may be automatically obtained from the smallest eigenvectors of the matting Laplacian. From the practical standpoint, matting components can help automate the matte extraction process and reduce user effort. Matting components

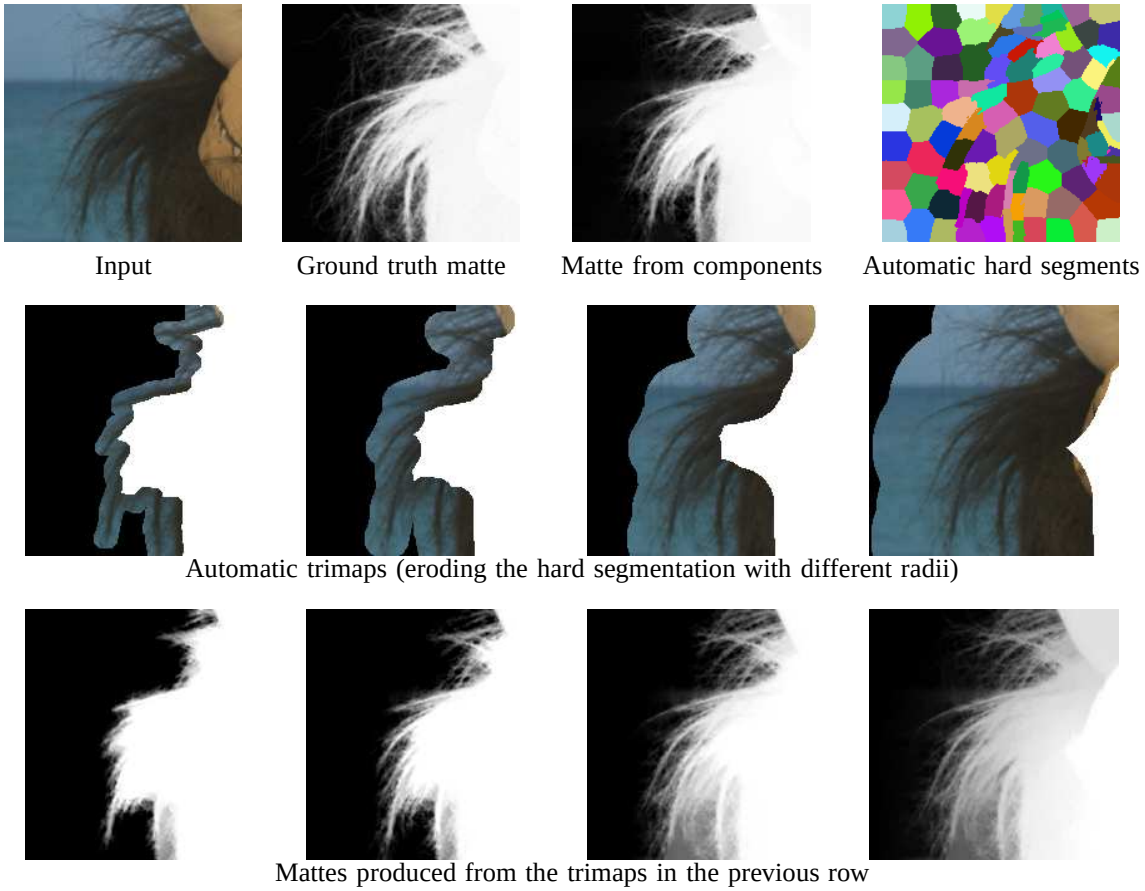


Fig. 10. Spectral matting vs. obtaining trimaps from a hard segmentation.

also suggest a new mode of user control over the extracted matte: while in previous methods the result is controlled by placement of hard constraints in image areas where the matte is either completely opaque or completely transparent, our new approach may provide the user with a simple intuitive preview of optional outputs, and thus enables the user to directly control the outcome in the fractional parts of the matte as well.

Limitations: Our method is most effective in automating the matte extraction process for images that consist of a modest number of visually distinct components. However, for highly cluttered images, component extraction proves to be a more challenging task. As an example, consider the case shown in Figure 11. The input image consists of a large number of small components. Projecting the ground truth matte (Figure 11a) on the subspace spanned by the 70 smallest eigenvectors results in a poor approximation (Figure 11b). Recall that since the matting components are obtained via a linear combination of the eigenvectors, they can do no better than the eigenvectors themselves, and thus Figure

input image

(a) (b) (c)

Fig. 11. Demonstration of limitations. Top: input image; Bottom: Ground truth matte (a); Mattes from 70 (b) and 400 (c) eigenvectors.

11b is the best matte that we could hope to construct from up to 70 matting components. Thus, it is quite clear that this number of components is insufficient to produce an accurate matte for this image. A better matte may be obtained from the 400 smallest eigenvectors (Figure 11c), but even this matte leaves room for improvement. We have not been able to test more than 400 eigenvectors due to computational limitations. We have empirically observed that this problem is significantly reduced if matting components are computed in local image windows independently, and are currently investigating methods for stitching together components obtained in different windows.

One major challenge in spectral matting is determining the appropriate number of matting components for a given image. This is a fundamental difficulty shared by all spectral segmentation methods. While the question of automatically selecting the number of components has been investigated (e.g. [27]), this parameter is still often manually adjusted. For the applications described in this paper we found that a useful strategy is to over-segment the image and group the components later using additional cues. A second free parameter in the algorithm is the number of smallest eigenvectors from which the components are formed (the number should be larger or equal to the number of components). In practice, we have observed that the performance is not very sensitive to this number and all results in this paper were obtained using the 70 smallest eigenvectors. Figure 12 demonstrates the effect of varying the number of matting components and the number of eigenvectors. It can be seen, that usually it is better to use a

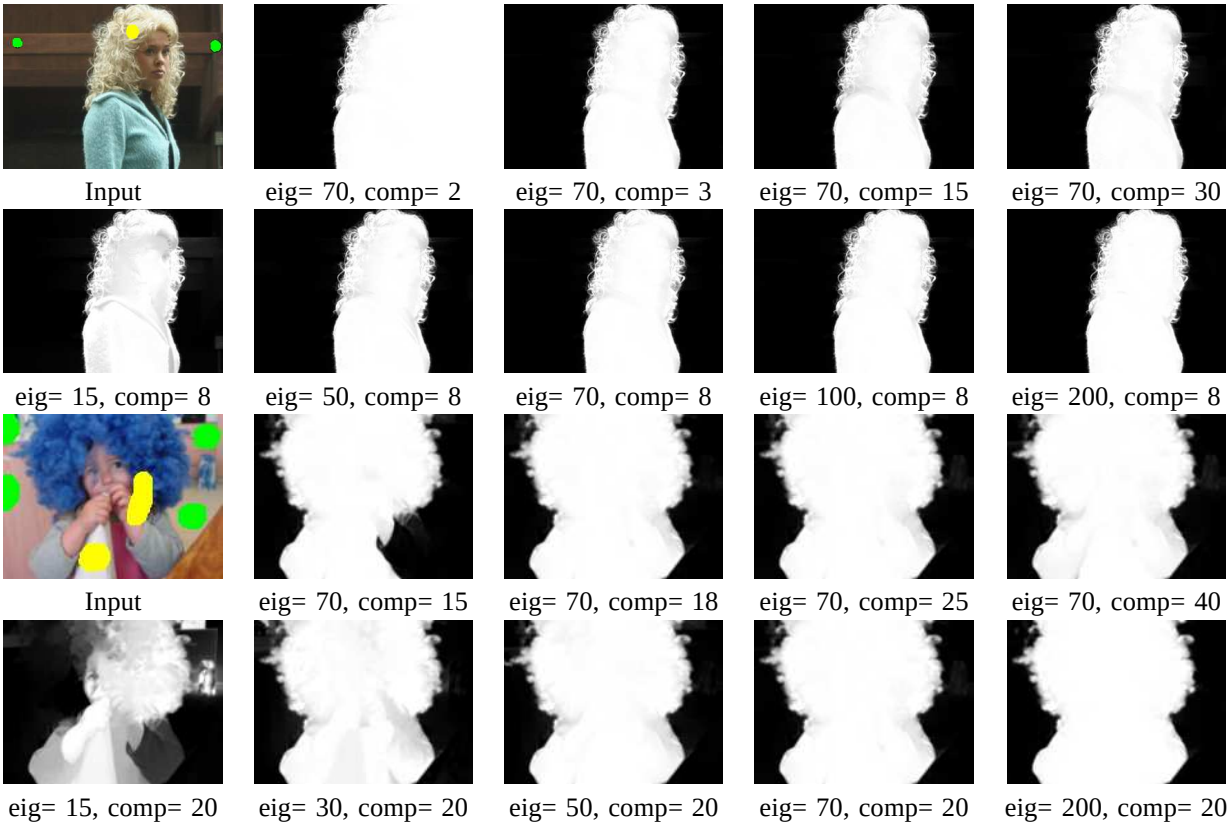


Fig. 12. The effect of changing the number of eigenvectors and the number of matting components on the resulting matte.

large number of matting components and a large number of eigenvectors, but this obviously makes the algorithm slower. When too many matting components are used, some regions in the alpha matte may become too transparent. This is due to some matting components that were not grouped correctly by the min-cut algorithm. The algorithm is much less sensitive to the number of eigenvectors being used. Using 50 to 200 eigenvectors usually produced similar results. It should be noted that increasing the number of eigenvectors makes the resulting alpha matte more opaque, as it gives more freedom to the non-linear optimization, which looks for opaque solutions. (In the extreme case, using all eigen-vectors will produce binary mattes).

Future directions: An important potential advantage of pre-segmenting the image into matting components is the option to compute meaningful color or texture histograms, or other statistics, within each component. The histogram similarity can provide another important cue to guide component grouping. This ability might significantly improve matting algorithms which make use of color models, such as [24]. For example, the current strategy in [24] is to build an initial color model using only the small number of pixels under the scribbles. This simple initialization is known to make the algorithm

sensitive to small shifts in the scribble location.

Given the growing interest in the matting problem and the large amount of recent matting research, it seems that an important future challenge is the design of an appropriate comparison between different user interaction modes and different matting algorithms. The ground truth data collected in this work is a step toward this goal, yet a proper user study is required in order to evaluate the amount of user time required for producing good results with each method.

Our code and ground truth data are available at: `www.vision.huji.ac.il/SpectralMatting`

REFERENCES

[1] P. A., S. H, and L. K-P. Partitoning sparse matrices with eigenvalues of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, 1990.

[2] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *Proc. ECCV*, pages 109–122, 2002.

[3] Y. Chuang, B. Curless, D. Salesin, and R. Szeliski. A bayesian approach to digital matting. In *Proc. CVPR*, pages 264–271, 2001.

[4] J. Dennis, J. Robert, and B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, 1983.

[5] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(98):298–305, 1973.

[6] M. Fiedler. Eigenvectors of acyclic matrices. *Czechoslovak Mathematical Journal*, 25(100):607–618, 1975.

[7] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25(100):619–633, 1975.

[8] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann. Random walks for interactive alpha-matting. In *Proc. VIIP*, pages 423–429, 2005.

[9] Y. Guan, W. Chen, X. Liang, Z. Ding, and Q. Peng. Easy matting: A stroke based approach for continuous image matting. *Computer Graphics Forum (Proc. Eurographics 2006)*, 25(3):567–576, 2006.

[10] S. Guattery and G. L. Miller. On the quality of spectral separators. *SIAM Journal on Matrix Analysis and Applications*, 19(3):701–719, 1998.

[11] H. K. An r-dimensional quadratic placement algorithm. *Management Science*, 17(3):219–229, 1970.

[12] V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts-a review. *IEEE Transactions on PAMI*, 29(7):1274–1279, 2007.

[13] K. Lang. Fixing two weaknesses of the spectral method. In *NIPS '05: Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.

[14] A. Levin, D. Lischinski, and Y. Weiss. A closed form solution to natural image matting. In *Proc. CVPR*, pages 61–68, 2006.

[15] A. Levin, A. Rav-Acha, and D. Lischinski. Spectral matting. In *CVPR*, June 2007.

[16] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Trans. Graph.*, 23(3):303–308, 2004.

[17] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*. MIT Press, 2001.

[18] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004.

[19] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on PAMI*, 22(8):888–905, 2000.

[20] A. Smith and J. Blinn. Blue screen matting. In *Proceedings of ACM Siggraph*, pages 259–268, 1996.

[21] J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum. Poisson matting. *ACM Trans. Graph.*, 23(3):315–321, 2004.

[22] D. Tolliver and G. Miller. Graph partitioning by spectral rounding: Applications in image segmentation and clustering. In *Proc. CVPR*, pages 1053–1060, 2006.

[23] D. W. and H. A. Algorithms for partitioning graphs and computer logic based on eigenvectors of connection matrices, 1973.

[24] J. Wang and M. Cohen. An iterative optimization approach for unified image segmentation and matting. In *Proc. ICCV*, pages 936–943, 2005.

[25] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *Proc. ICCV*, pages 975–982, 1999.

[26] S. X. Yu and J. Shi. Multiclass spectral clustering. In *Proc. ICCV*, pages 313–319, 2003.

[27] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *NIPS*. MIT Press, 2005.