Visual Information Processing Laboratory

School of Computer Science and Information Technology

The University of Nottingham

Technical Report

Report-VIPLAB-01-2006

January 2006

Interactive Image Matting using Optimization

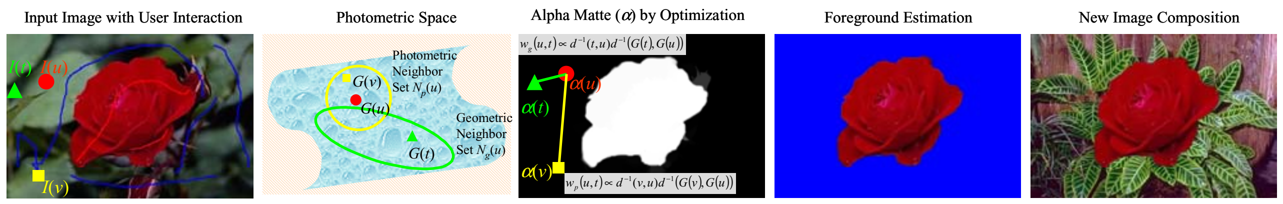
Jian Guan and Guoping Qiu

School of Computer Science and Information Technology
The University of Nottingham

Interactive Image Matting using Optimization

Jian Guan and Guoping Qiu

School of Computer Science and Information Technology, University of Nottingham


Input Image with User Interaction
$I(t)$
$I(u)$
$I(v)$
Photometric Space
Photometric Neighbor Set $N_p(u)$
$G(v)$
$G(u)$
$G(t)$
Geometric Neighbor Set $N_g(u)$
Alpha Matte (α) by Optimization
$w_g(u,t) \propto d^{-1}(t,u)d^{-1}(G(t),G(u))$
$\alpha(u)$
$\alpha(t)$
$\alpha(v)$
$w_p(u,t) \propto d^{-1}(v,u)d^{-1}(G(v),G(u))$
Foreground Estimation
New Image Composition

Blue scribbles: $\alpha(u_i) = $ "*bg*"
Red scribbles: $\alpha(u_j) = $ "*fg*"

$$\alpha = \arg_{\forall\alpha}\left\{\min \sum_u \left(\alpha(u) - \sum_{t\in N_g(u)} w_g(u,t)\alpha(t) - \lambda \sum_{v\in N_p(u)} w_p(u,v)\alpha(v)\right)^2 \quad \text{subject to} \quad \alpha(u_i) = \text{"}bg\text{"} \quad \alpha(u_j) = \text{"}fg\text{"}\right\}$$

Figure 1: For a given image we want to segment the foreground/background and compute its alpha matte. Our one-step segmentation and matting algorithm solves for the alpha matte by optimizing a quadratic matting cost function subject to user input constraints. From left to right: (a) Input image with user indicated background and foreground pixels. (b) Projecting the pixels onto their photometric feature space where geometrically (spatially) adjacent pixels are not necessarily close to each other and spatially far away pixels may be very close to each other. (c) Pixels of the alpha matte are linked to their geometric and photometric neighbors by the strength of their pixels photometric similarities, and the alpha matte is obtained by minimizing a quadratic matting cost function defined by the links of the *matting neighbors*. (d) After obtaining the alpha matte, the foreground of the input image can be estimated; here the obtained foreground is displayed against a uniform background. (e) The alpha matte can also be used to composite a new image with a complex new background.

Abstract

In this paper, we formulate interactive image matting as a constrained optimization problem. We first make some simple and reasonable assumptions about the alpha matte and assume that, geometrically, the closer two pixels are, the more likely they will have similar alpha values, conversely, the farther apart two pixels are, the more likely they will have different alpha values; photometrically, the more similar two pixels are, the more likely they will have similar alpha values, conversely, the more different two pixels are, the more likely they will have different alpha values. We then formulate these assumptions in a quadratic matting cost function and obtain the alpha matte by minimizing the matting cost function. User interaction in the form of a few scribbles indicating a few definite background and foreground pixels is used to provide constraints to make the problem well posed. For a given set of constraints the matting cost function has a unique global minimum and can be solved efficiently using standard methods. With the computed alpha matte we then estimate the background and foreground pixels. Results show that the new method works effectively and provides an alternative computational algorithm for building interactive image editing tools.

Keywords: interactive image segmentation, alpha matting, constrained optimization.

1. Introduction

Interactive background foreground segmentation and image matting are two closely related problems that have wide applications in computer graphics. The key feature of recent methods [Boykov and Jolly 2001, Chuang et al. 2001, Li et al. 2004, Rother et al. 2004 Sun et al. 2004, Wang and Cohen 2005] is that they all consist of two main parts, the user interaction part and the computational part. These two parts are closely linked, on one hand, user interaction provides strong constraints to make the computational problem well conditioned, and on the other hand, the computational methods determine the amount of user labor required to generate a good matte. Some methods require user input in the form of a well-drawn trimap [Chuang et al. 2001, Sun et al. 2004] and others require user effort in the form of a few scribbles [Boykov and Jolly 2001, Li et al. 2004, Rother et al. 2004, Wang and Cohen 2005]. The computational algorithms used in previous interactive image segmentation and image matting work can be classified into four main categories, Graph Cut [Boykov and Jolly 2001, Li et al. 2004, Rother et al. 2004], Bayesian [Ruzon and Tomasi 2000, Chuang et al. 2001], Poisson Equation [Sun et al. 2004] and Belief Propagation [Wang and Cohen 2005].

This paper presents a new interactive algorithm for image foreground background separation and matting. Our method also requires minimal user interaction in the form of a few scribbles, but we introduce an alternative computational algorithm to compute a good alpha matte. We formulate image matting as a constrained optimization problem. We define a quadratic form matting cost function based on the assumptions that, geometric or spatial neighbors are likely to have similar alpha values, photometrically similar pixels are likely to have similar alpha values, and the similarity (dissimilarity) in alpha values are related to their geometric closeness and photometric similarity. We employ the user input as constraints to make the problem well posed and the matting cost function has a unique global minimum. We obtain the alpha matte by minimizing the matting cost function, which yields a large, sparse system of linear equations that can be solved using standard numerical computational tools.

2. Related Work

In this section, we give a very brief review of related recent literature in interactive image segmentation, image matting, and unified image segmentation and matting.

Interactive image segmentation: Image segmentation has been studied for a long time and interactive approach have recently attracted increasing interest because it can help overcome the inherent difficulties of automatic image segmentation. The interactive graph cut method of [Boykov and Jolly 2001] and the segmentation given partial grouping constraints method of [Yu and Shi 2004] are representatives of recent progress. The GrabCut method of [Rother et al. 2004] and the Lazy Snapping system of [Li et al.] have extended the graph cut algorithm of [Boykov and Jolly 2001] to develop simpler and easier to use image editing tools.

Image Matting: Software systems for image matting include Magic Wand [Adobe 2002] and Knockout [Corel Corporation 2002]. Recent computational approaches to image matting include the Bayesian approach of [Chuang et al. 2001] which was based on [Ruzon and Tomasi 2000] and Poisson Matting [Sun et al. 2004]. These methods use a well-drawn trimap to obtain a good matte. The GrabCut method [Rother et al. 2004] which was designed to simplify user interaction uses graph cut to segment the background and foreground first and then performs border matting in the border areas between segmented foreground background regions.

Integrated Image Segmentation and Matting: A recent work that performs interactive image segmentation and image matting was introduced in [Wang and Cohen 2005] where they unified image segmentation and matting and generated a matte directly from a few user specified foreground and background strokes. They proposed a Belief Propagation solution to iteratively estimate a discrete (25 levels) matte.

3. Algorithm

The image composition and matting problem can be summarized by following equation:

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

where u denotes the pixel co-ordinate vector (x, y), $B(u)$ denotes the background image and $F(u)$ denotes the foreground image and $\alpha(u) \in [0, 1]$ denotes the alpha matte image used to linearly blend between foreground and background

In the case of matting, or pulling of matte, the image $I(u)$ is given, the task is to find the alpha image $\alpha(u) \in [0, 1]$, the background image $B(u)$ and the foreground image $F(u)$ such that the given image $I(u)$ can be modeled as a linear combination of $B(u)$ and $F(u)$ by $\alpha(u)$.

It is well known that matting is an under constrained problem and user interaction has been previously used to compute good quality mattes, see e.g. [Rother et al. 2004, Sun et al. 2004, Wang and Cohen 2005]. We have developed a new computational approach to interactive image matting. Our method also requires minimal user interaction and we introduce a new alternative computational method to generate a good quality matte.

3.1. Algorithm Overview

We first define a quadratic matting cost function of $\alpha(u)$ based on some simple and reasonable assumptions. We assume that, geometrically or spatially, the closer two pixels are, the more likely they will have similar alpha values; conversely, the farther apart two pixels are, the more likely they will have different alpha values; photometrically, the more similar two pixels are, e.g., similar colors, similar texture neighborhoods, the more likely they will have similar alpha values; conversely, the more different two pixels are, the more likely they will have different alpha values. Image matting is formulated as an optimization problem and the task becomes that of minimizing the deterministic quadratic matting cost function.

We then use an interactive approach [Boykov and Jolly 2001, Rother et al 2004, Wang and Cohen 2005] to solve the (under constrained) optimization problem. A user scribbles on the image indicating certain pixels that are definitely background, i.e., $\alpha(u) = 0$ and certain pixels that are definitely foreground, i.e., $\alpha(u) = 1$. Using these user inputs as constraints makes the problem well posed and the matting cost function has a unique global minimum. The optimization problem yields a large, sparse system of linear equations, which can be solved efficiently using a number of standard methods.

The deterministic global solution to the quadratic matting cost function yields a continuous alpha matte $\alpha(u) \in [0, 1]$ which can be used directly to estimate the background and foreground pixels. If $\alpha(u) = 0$ then it corresponds to a definite background pixel $B(u) = I(u)$. If $\alpha(u) = 1$, then it corresponds to a definite foreground pixel $F(u) = I(u)$. If $0 < \alpha(u) < 1$, then within a neighborhood of u, $N(u)$, we find $\forall s \in N(u)$ and $\alpha(s) = 0$; $\forall t \in N(u)$ and $\alpha(t) = 1$, and estimate $B(u)$ and $F(u)$ from the set $\{B(s), F(t)\}$ using a method similar to that of [Wang and Cohen 2005]:

In summary, our algorithm starts from the definition of a deterministic quadratic matting cost function, and then with the aid of user interaction, it then solves a large, sparse system of linear equations using standard numerical methods to obtain a continuous alpha matte, and finally, it uses the alpha matte to estimate the background and foreground pixels.

3.2. Matting as Constrained Optimization

Let $G(u) = \Phi(I(u))$ be image features (such as color, texture, etc), computed around the pixel at location u, where Φ is the feature extraction operator. We call $G(u)$ the photometric features. Let $N_g(u)$ be the set that contains the *geometric* neighbors of u, then

$v \in N_g(u)$ if $|v - u| < R_g$, let $N_p(u)$ be the set that contains the *photometric* neighbors of u, if $|G(v) - G(u)| < R_p$ then $v \in N_p(u)$, where R_g and R_p are some preset constants determining the size of the neighborhoods. Let $N_m(u) = N_g(u) \cup N_p(u)$ denote the set that contains the *matting neighbors* of u and which includes all pixels that are either the geometric neighbors or the photometric neighbors of u. Note that if the pixel $I(v)$ is a geometric neighbor of the pixel $I(u)$, $I(v)$ can also be the photometric neighbor of $I(u)$, but if $I(v)$ is a photometric neighbor of $I(u)$, $I(v)$ may not necessarily be the geometric neighbor of $I(u)$.

Based on some reasonable assumptions about the alpha matte and assuming that two spatially adjacent pixels should have similar alpha values, two photometrically similar pixels should also have similar alpha values, and the difference in alpha values between two pixels should be proportional to the pixels' photometric distance, the alpha matte may be obtained by minimizing the matting cost function defined as

$$E(\alpha(u)) = \sum_u \left(\alpha(u) - \sum_{v \in N_m(u)} \left(w_g(u,v) + \lambda w_p(u,v) \right) \alpha(v) \right)^2 \qquad (2)$$

where $w_g(u, v)$ is the geometric neighbor similarity weighting function between two pixels u and v, $w_p(u, v)$ is the photometric neighbor similarity weighting function between two pixels u and v, and λ is a constant that measures the relative importance of the geometric neighbor similarity and the photometric neighbor similarity, all similarity weightings inside a matting neighborhood sum to one.

$$\sum_{v \in N_m(u)} \left(w_g(u,v) + \lambda w_p(u,v) \right) = 1 \qquad (3)$$

To reflect the assumptions that geometrically close pixels are likely to have similar alpha values and geometrically far apart pixels are likely to have different alpha values, and photometrically similar pixels are likely to have similar alpha values and photometrically different pixels are likely to have different alpha values, the neighbor similarity functions $w_p(u, v)$ and $w_g(u, v)$ should be small if $|u - v|$ is large and small if $|u - v|$ is large, and they should be small if $G(u)$ and $G(v)$ are different and large if $G(u)$ and $G(v)$ are similar. We use a function that has been used in the image segmentation literature, e.g., [Yu and Shi 2004]

$$\begin{aligned} &\text{if } I(v) \in N_g(u) \quad w_g(u,v) = e^{-\|v-u\|^2/\sigma_{gg}^2} e^{-\|G(v)-G(u)\|^2/\sigma_{gp}^2} \\ &\text{if } I(v) \in N_p(u) \quad w_p(u,v) = e^{-\|v-u\|^2/\sigma_{pg}^2} e^{-\|G(v)-G(u)\|^2/\sigma_{pp}^2} \end{aligned} \qquad (4)$$

where σ_{gg} and σ_{gp}, σ_{pg} and σ_{pp} are the variances of the geometric co-ordinates and photometric features inside $N_p(u)$ and $N_g(u)$ respectively.

With user interaction, a set of alpha matte pixels at locations u_i are specified as background ($\alpha(u_i) = 0$) and as foreground ($\alpha(u_i) = 1$). We minimize the matting cost function $E(\alpha(u))$ in (2) subject to these constraints. The cost function is quadratic and the constraints are linear, the cost function has a unique global minimum. This optimization problem yields a large, sparse system of linear equations, which may be solved efficiently using a number of standard tools [Hackbusch 1985, Press et al. 2002]. The colorization work of [Levin et al. 2004], the machine learning work of [Roweis and Saul 2000] and normalized cut of [Shi and Malik 2000] solve a similar optimization problem.

3.3. Implementation Procedure

The implementation of our matting algorithm consists of several straightforward steps. First we need to decide the types of photometric features to use and find the photometric neighbors for each pixel, and then we compute $\alpha(u)$ by optimizing $E(\alpha(u))$ using user input as constraints, which is achieved by solving a large, sparse system of linear equations. With the computed alpha matte, we estimate the background and foreground pixels for image matting or segmentation. See Figure 1 for the work-flow of our algorithm.

Step 1: *Selecting Matting neighbors*

The selection of geometric neighbors is straightforward and in all our experiment we use the 8 spatially connected pixels as geometric neighbors.

We then need to select the photometric features for finding the photometric neighbors and for computing the similarity weighting measures (4) in the feature space. In all our experiment, we convert the color image into *La*b** color space [CIE 1986] because it has been shown that the color distance in this space corresponds well with perceptual difference. Although there are many possibilities to extract compact features, we simply use all pixels from the 3x3 window that center on a pixel to form the photometric feature for that pixel. Therefore, the photometric feature of a pixel at location u, $G(I(u))$, is a 27-d vector (9 pixels x 3 color channels inside a 3x3 window centered on u). Note that other features, for example, multi-channel texture features [Bovick et al. 1990], can also be used, which may be helpful in difficult cases such as separating highly textured foreground and background regions.

As will be discussed later in Section 3.4, we cannot and should not have too many photometric neighbors for each pixel. Fortunately, it turns out that it is only necessary to have several photometric neighbors for each pixel (we use 4 in all our experiments). We use a very simple method to efficiently search the photometric neighbors. For each pixel, we randomly sample K pixels from an M x N window center on the pixel[1]. From these K pixels, we find 4 that are the closest to the pixel as its photometric neighbors. Each pixel will have 8 geometric neighbors and 4 photometric neighbors. Therefore each alpha pixel is linked to 12 (or fewer) matting neighbors via their similarity weightings (4). The relative geometric and photometric weighting constant λ is determined by experiment, which may vary from image to image. We found setting $\lambda = 1$ works well for most images.

Step 2: *Solving constrained optimization problem*

[1] In theory, we should sample as large a window and as many pixels as possible, at the extreme, sample the whole image and include all pixels, but that will be more time consuming and turns out not necessary since all matting neighbors are linked. In practice, we only need to sample a small region and a small number of random samples. In all our experiments, w use $K = 150$ and M x N = 17 x 17, which work well for all images we tested.

Solving the minimization problem of (2) using the user inputs as constraints yields a large, sparse system of linear equations of the form $\boldsymbol{W}\boldsymbol{a} = \boldsymbol{c}$, where $\boldsymbol{W}$ is a sparse matrix containing the connection weights of matting neighbor pixels, $\boldsymbol{a} = \{\alpha(u)\}$ is the vector containing all alpha pixels, and $\boldsymbol{c}$ is a constant vector. In our current implementation, we use Matlab's build-in least squares solver for sparse linear system to directly solve for $\alpha(u)$. On a Pentium 4 PC with 1.8GHz CPU, our current implementation takes about 6 minutes to compute a matte for a 500x300 image. However, fast methods and even dedicated hardware are available to solve this problem much faster [Backbusch 1985, Press et al. 2002, Geiselmann et al. 2005].

Step 3: *Estimating Background and Foreground*

After obtaining $\alpha(u)$, we can now find $F(u)$ and $B(u)$ in (1). If $\alpha(u) = 1$, then $F(u) = I(u)$. If $\alpha(u) = 0$, then $B(u) = I(u)$. If $0 < \alpha(u) < 1$, then within a neighborhood $N(u)$ (for which we use a 17 x 17 window centered around u in all our experiments), we find $\forall s \in N(u)$ and $\alpha(s) = 0$ and $\forall t \in N(u)$ and $\alpha(t) = 1$, and then estimate $B(u)$ and $F(u)$ using a method similar to that of [Wang and Cohen 2005]:

$$\left[F(u), B(u)\right] = \arg\min_{\forall s,t}\left(I(u) - \alpha(u)F(t) - (1-\alpha(u))B(s)\right) \quad (5)$$

3.4. Algorithm Analysis

Unlike previous methods [Chuang et al. 2001, Boykov and Jolly 2001, Rother et al. 2004, Wang and Cohen 2005], in our solution, the user marked background foreground pixels are not explicitly modeled or directly used to match the pixels, but rather, these user provided seed pixels are used as constraints to solve a quadratic cost function. The user inputs are implicitly propagated to the whole image by the optimization process.

It is important to note that even though the pixels are only linked to a few of their matting neighbors, the process of optimizing (2) computes all alpha values *together* by solving a large, sparse linear system of equations, a global operation that couples all alpha pixels that lie in the same connected component of the graph defined by the matting neighbors. The algorithm incorporates the user input to find a globally coherence alpha matte by integrating information from overlapping neighbors.

Note also that we divide the neighbors of a pixel into two categories – geometric neighbors and photometric neighbors. The reasons that it is necessary to have these two types of neighbors are as follows.

The value of $\alpha(u)$ is influenced by it's geometric neighbors (spatially adjacent pixels) $\alpha(v)$ to ensure that the alpha matte is smooth. Using a weighting function in (4) we ensure that if $G(u)$ and $G(v)$ are similar, then $\alpha(u)$ and $\alpha(v)$ should be similar also. However, if $G(u)$ and $G(v)$ differ significantly, then $\alpha(u)$ is not affected by $\alpha(v)$ even though they are spatial neighbors. This is to ensure that the alpha matte is not overly smooth especially in areas where the image changes significantly, because a large change in the image may signify the changes from background to foreground, or vice versa, therefore the value of alpha should change accordingly.

The value of $\alpha(u)$ is also influenced by its photometric neighbors. Two pixels having similar photometric features would be very likely belonging to the same background or the same foreground even though they may be very far away spatially. Using a weighting function in (4) we ensure that if $G(u)$ and $G(v)$ are photometrically very similar, even though they may be geometrically very far apart, minimizing $E(\alpha(u))$ in (2) will favor $\alpha(u)$ and $\alpha(v)$ have similar values.

In an interactive approach to foreground background separation, the introduction of the photometric neighbors will greatly simplify user input as illustrated in Figure 2. If only geometric neighbors are used (geometric neighborhoods are usually small, typical sizes are 3 x 3 or 5 x 5, we will discuss shortly why we cannot have large geometric neighborhoods), then the user input will have to be placed in every hole of the net to indicate the background because the links between holes would have been cut off by the net hence the user input will not be able to propagate from one hole to the next. On the other hand, the photometric features inside all the holes are very similar and they will be very likely to be photometric neighbors. Therefore, it is only necessary to put one stroke inside one hole and this user input will be propagated through to all the holes via photometric neighbors. In this way, user effort is greatly reduced while achieving the same good quality matting.

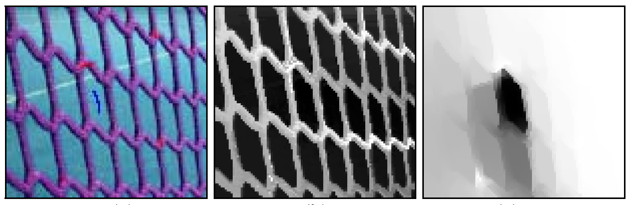


Figure 2: (a) A net image with many holes, the blue scribble indicates background, the red scribbles indicate foreground. (b) The alpha matte pulled with photometric neighbors and (c) The alpha matte pulled without the use of photometric neighbors (note that the blue stroke failed to propagate outside of its hole).

At a first glance, it may seem that separating pixels into geometric neighbors from photometric neighbors is unnecessary if one only wants to bring spatially far away pixels to have influence on each other's alpha values. Indeed, if this were the case, all we have to do is to make the geometric neighborhood larger (at the extreme to include the whole image). However, this will have two problems. Firstly, a larger neighborhood will make the problem less sparse thus making the problem computationally much more demanding. Secondly, making a pixel influenced by too many pixels (with unknown status), it may also risk introducing inaccuracy and uncertainty into the model.

Using the concept of photometric neighborhood, we can select only those pixels that are very similar to each other to have an influence on each other's alpha value. In this way, we can cover a much larger spatial area and at the same time still ensure the problem is sparse thus can be solved efficiently using standard numerical tools. Fortunately, our experiences show that only using a few photometric neighbors (about 4) suffice to ensure good results. With the concept of photometric neighborhood, we can also ensure that if two pixels are spatially far away from each other, then in order to have an influence on each other's alpha values, they have to be photometrically very similar. In other

Algorithms	Trimap Requirement	Cost Function/Model	Computational Method	Alpha Values
Graph Cut[1]	Simple/Interactive	Deterministic	One shot/Graph cut	Binary
GrabCut[2]	Simple/Interactive	Statistical/GMM	Iterative/Graph cut	Binary/Continuous
Bayesian[3]	Complex	Statistical/GMM	Iterative/MAP	Continuous
Poisson Matting[4]	Complex	Deterministic	Poisson Equation	Continuous
Wang Cohen[5]	Simple/Interactive	Statistical/MRF/GMM	Iterative/Belief propagation	Discrete (25levels)
Ours	Simple/Interactive	Deterministic Quadratic	One shot/Linear equations	Continuous
[1][Boykov and Jolly 2001] [2][Rother et al 2004] [3][Chuang et al 2001] [4][Sun et al 2004] [5][Wang and Cohen 2005]				

Table 1: A comparison of typical features of several state of the art foreground background segmenting and image matting algorithms.

words, we can ensure that only those pixels that are highly likely to belong to the same background or foreground to have an influence each other's alpha values, therefore reducing the uncertainty in the model. We consider the photometric neighborhood concept and implementation procedure a noteworthy feature of our algorithm.

Table 1 shows a comparison of the features of our algorithm and those of several state of the art methods. We did not show computational speeds in the table as these may depend on implementation details. In terms of complexity, graph cut has a low order polynomial complexity, however, it can be computed very efficiently and therefore the algorithm can be considered fast [Boykov and Kolmogorov 2004]. Belief Propagation is slow because of its iterative nature but [Wang and Cohen 2005] reported that dramatic speed up was possible with fast algorithms. Our algorithm solves a large, sparse linear system of equations, which when implemented using advanced techniques [Hackbusch 1985, Press et al. 2002] has a complexity scales linearly with the number of pixels. It is interesting to note also that recently, special hardware has been developed to solve large, sparse linear system of equations [Geiselmann et al. 2005].

4. Results

We have tested our method on a variety of images. Figure 3 shows some typical examples of the interaction process of our algorithm. Normally, an initial set of scribbles will miss some small details, then the user can add a few more scribbles and the result normally comes out satisfactorily in 1 – 2 rounds of interaction. In some more complex cases, for example, if the foreground and background contain textured patterns, a few interactions may be necessary. It is seen that these images contain some very complicated textures, the tiger's skin, for example, contain different colors and texture patterns. It is seen that our technique works satisfactorily.

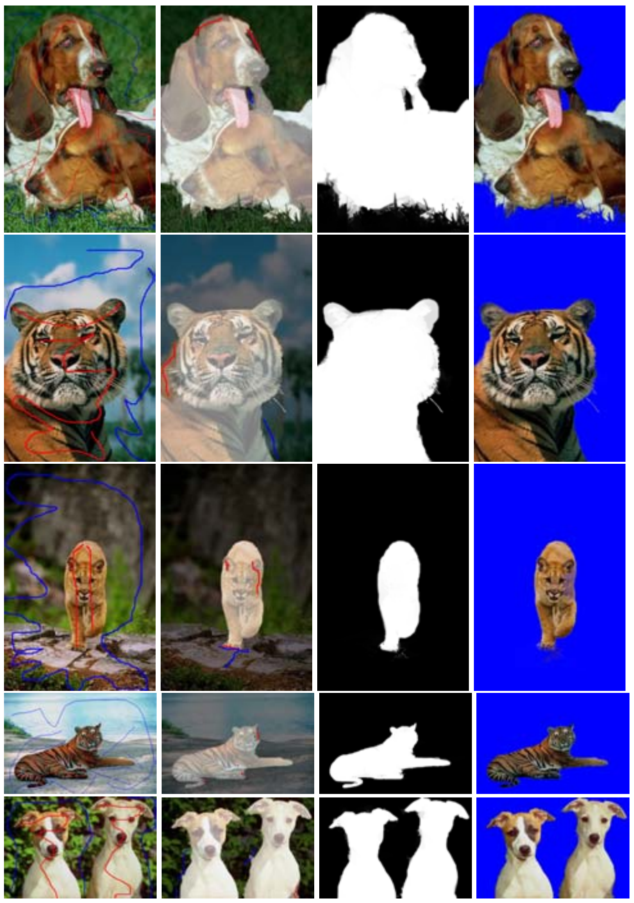


Figure 3: 1st column, original image with initial scribbles put on by the user indicating foreground (red) and background (blue). 2nd column, user interaction. 3rd column, alpha matte. 4th column the foreground images viewed against a uniform blue background.

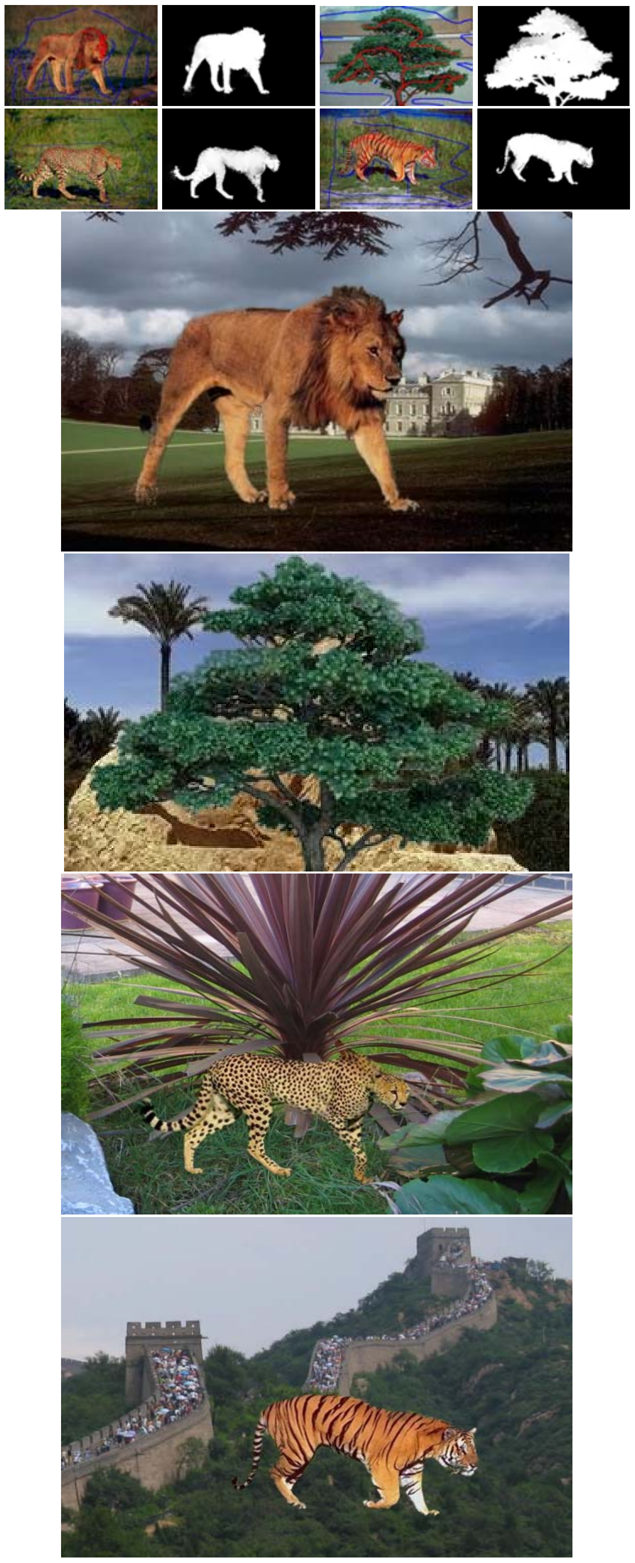


Figure 4: Examples of image composition. In each case, the input image with the total scribbles put on by the user to extract the matte, the final matte and the new composite image formed with the matte and a new background is shown.

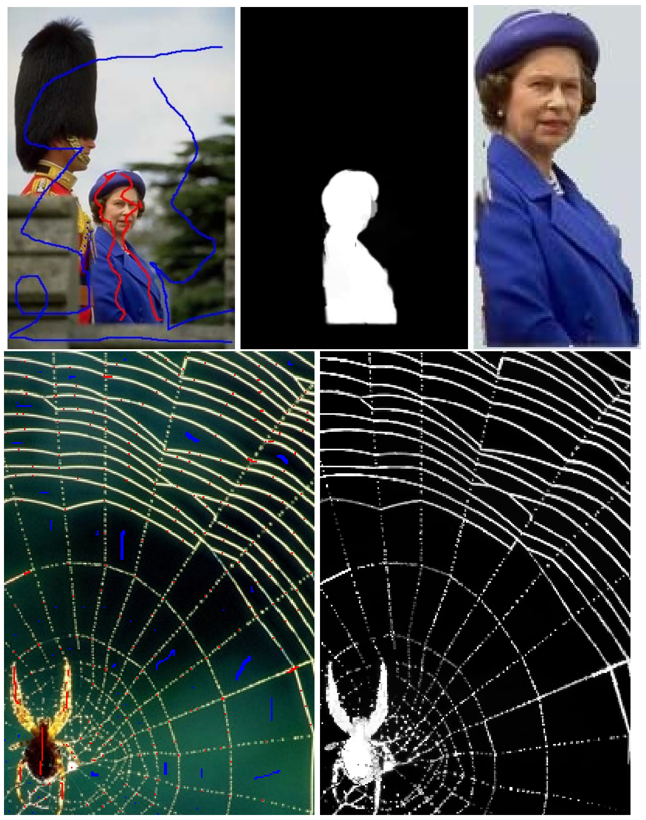


Figure 5: Top row, here we show that our method is capable of removing unwanted objects from the image, also shown is the extracted matte and the estimated foreground. Bottom row, here is a difficult case to draw a good trimap, our method has succeeded in pulling out the spider web.

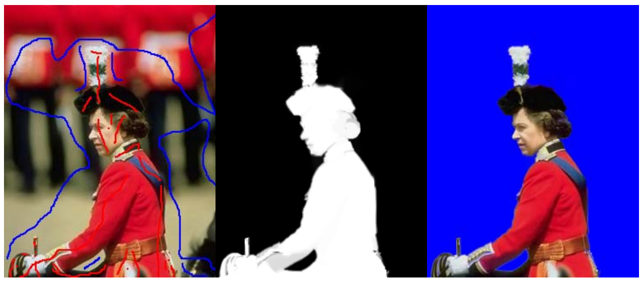


Figure 6: Our method copes with confusing background and foreground colors very well. In this example, the red coat of the Queen and the red color on top of the image are very similar. Especially pay attention to the white Plume the Queen was wearing and the white colors surrounding the white Plume. Our technique has successfully separated these similar color foreground and background.

Figure 4 shows examples of image composition using our interactive segmentation and matting algorithm. We first estimated the alpha matte and then combined the extracted foreground with a new background to form a new image. Again the results are satisfactory.

Figure 5 shows that our method is capable of removing unwanted complicated objects from the image and pulling out complicated features such as a spider's web with low to moderate user effort. More examples of complex foreground extraction are shown in Figure 7.

Figure 6 shows that our method is capable of separating confusing foreground and background with similar colors. In statistical based models, [Wang and Cohen 2005] noticed that this could cause a problem. Because our method not only taking the photometric features (colors in this case) of the pixels into consideration but also explicitly modeling their geometric locations, it has succeeded in separating two similar colors located in different places.

Figure 8 shows the interactive process of extracting the foreground of an image and compared with the GrabCut result. Figure 9 shows another of our result as compared with Bayesian matting, Graph Cut and GrabCut. It is seen that our results are comparable with those of state of the art methods.

5. Summary

In this paper, we have presented a new interactive algorithm for foreground background segmentation and for image matting. Based on a set of reasonable assumptions about the alpha matte, we formulated the matting problem as a constrained optimization problem and presented a method to construct the matting cost function and solve the optimization problem. We have presented results which show that our method works effectively and is competitive to state of the art methods. We also show that our method can cope with difficult cases that existing algorithms may fail. In conclusion, we have presented an alternative approach to interactive image segmentation and matting which may provide a building block for constructing powerful image editing tools.

6. References

ADOBE SYSTEMS INCORP. 2002. Adobe Photoshop User Guide.

HACKBUSCH, W. 1985. Multi-grid Methods and Applications, *Springer*, Berlin, 1985

BOVICK, A. CLARK, M. AND GEISLER. W. 1990. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1), January 1990

BOYKOV, Y., AND JOLLY, M. P. 2001. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *Proc. IEEE Int. Conf. on Computer Vision*

BOYKOV, Y AND KOLMOGOROV, V. 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on PAMI*, Vol. 26, 1124-1137

CIE 1986. Colorimetry, CIE Pub. No. 15.2, Centr. Bureau CIE, Vienna

CHUANG, Y.-Y., CURLESS, B., SALESIN, D., AND SZELISKI, R. 2001. A Bayesian approach to digital matting. In *Proc. IEEE CVPR 2001*, CD-ROM.

COREL CORPORATION, 2002. Knockout user guide.

GEISELMANN, W., SHAMIR, A., STEINWANDT, R., AND TROMER, E. 2005. Scalable hardware for sparse systems of linear equations, with applications to integer factorization. *Proc. CHES 2005, LNCS 3659*, 131--146, Springer

LEVIN, A., LISCHINSKI, D. AND WEISS, Y. 2004. Colorization using Optimization. *SIGGRAPH, ACM Transactions on Graphics*, 689-694

LI, Y., SUN, J., TANG, C-K. AND SHUM, H-Y. 2004. Lazy Snapping. *ACM Transaction on Graphics, Vol 23, No. 3*, 303-308

PRESS, W. H. AND TEUKOLSKY, S. A. ET AL. 2002. *Numerical Recipes in C++ - The Art of Scientific Computing*, Cambridge, MA, Cambridge University Press, 2002

ROWEIS, S AND SAUL, L. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science*, v.290, no.5500 , 2323-2326

ROTHER, C., KOLMOGOROV, V., AND BLAKE, A. 2004. "GrabCut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph*. 23(3): 309-314

RUZON, M. AND TOMASI, C. 2000. Alpha estimation in natural images. In *Proc. of IEEE CVPR 2000*, 18–25

SHI, J. AND MALIK, J. 2000. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 888-905

SUN, J., JIA, J.Y., TANG, C.K. AND SHUM, H.Y. 2004. Poisson matting. *Proc. of ACM SIGGRAPH*, pp. 315-321

WANG, J. AND COHEN, M., 2005. An iterative optimization approach for unified image segmentation and matting. In *Proc. IEEE Int. Conf. On Computer Vision*

YU, S.-X. AND SHI, J. 2004. Segmentation given partial grouping Constraints. *IEEE TPAMI*, 173–183

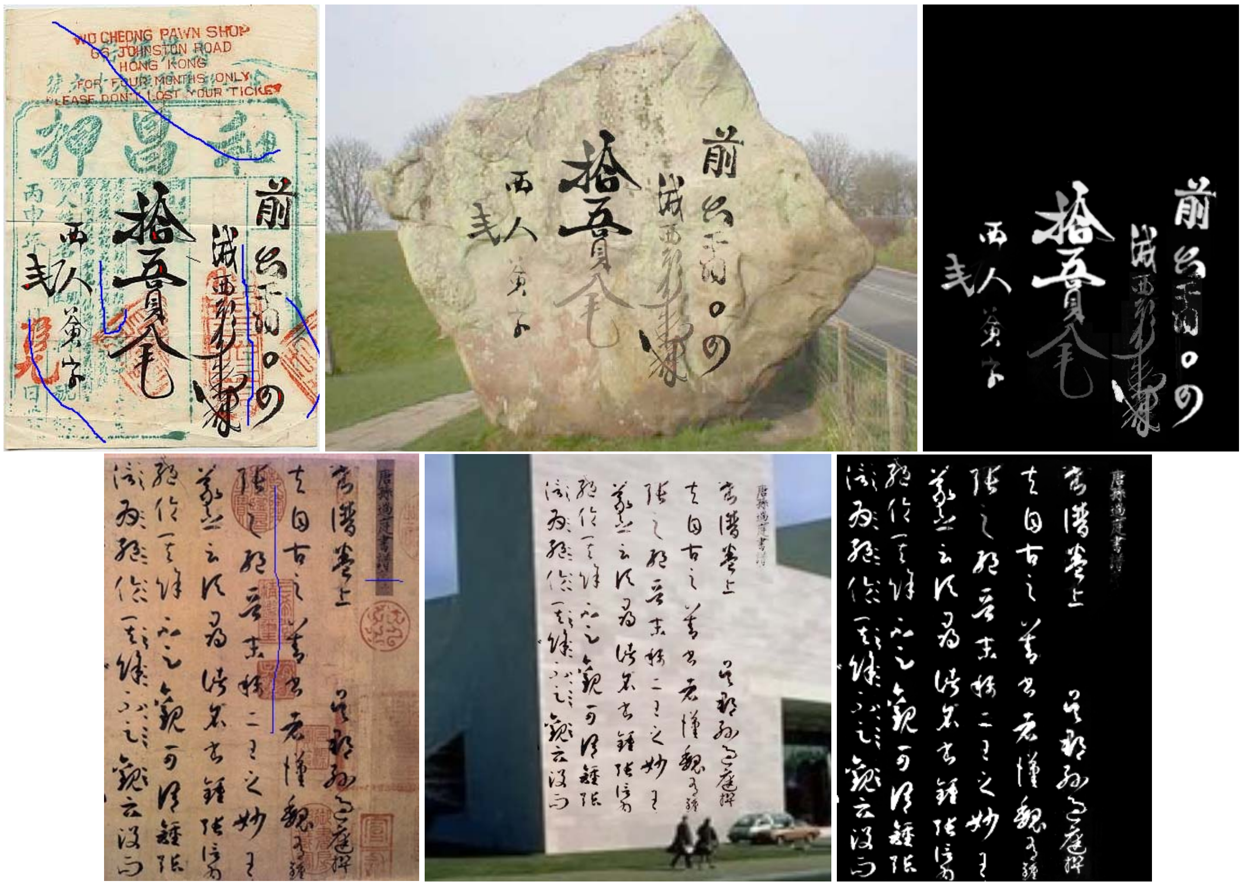


Figure 7: More examples of extracting complex patterns from image using our method. The left images show the original with user indicated foreground (red) and background (blue). The right images are extracted alpha matte. The middle images show the composition of the extracted foregrounds with new backgrounds.

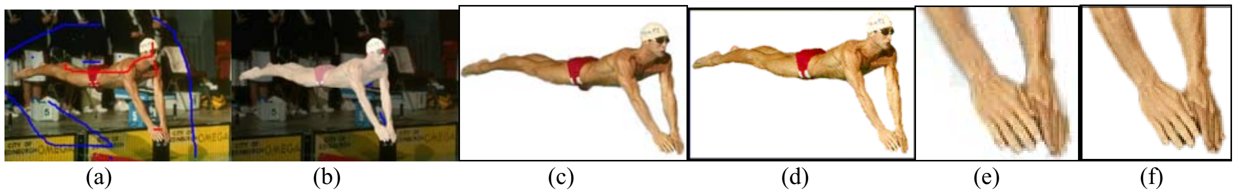


Figure 8: A comparison with GrabCut [Rother et al. 2004]. (a) Original image with initial scribbles. (b) 2nd interaction based on the result from (a). (c) Foreground image extracted by our algorithm. (d) Result of GrabCut. (e) Zoomed sub-image of (c) and (f) zoomed sub-image of (d). **Acknowledgement**: This image was used in [Rother et al 2004], the original image was downloaded from Berkeley Image Database[2]. The results of GrabCut were cutouts from the electronic paper of [Rother et al. 2004].

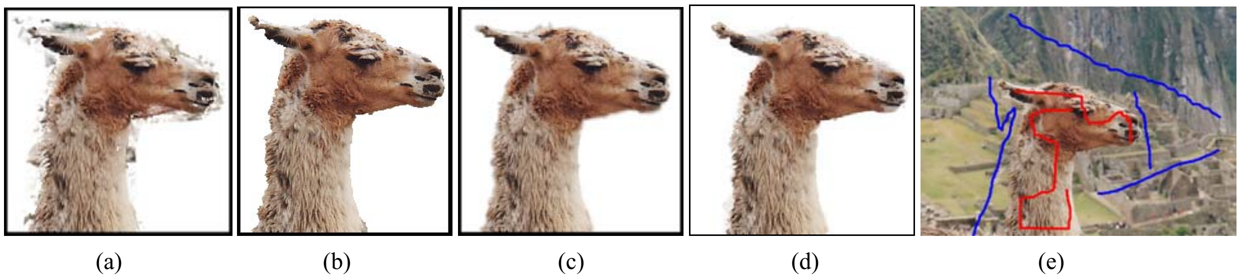


Figure 9: A comparison of results. (a) Bayesian Matte. (b) Graph Cut. (c) GrabCut. (d) Our result. (e) The original image with initial user input to generate (d). **Acknowledgement**: (a) to (c) are direct cutouts from the electronic paper of [Rother et al 2004] and the original image was down loaded from: http://research.microsoft.com/vision/cambridge/i3l/segmentation/GrabCut.htm.

[2] http://www.cs.berkeley.edu/projects/vision/grouping/segbench/BSDS300-images.tgz