User-friendly Interactive Image Segmentation through Unified Combinatorial User Inputs

Wenxian Yang, Jianfei Cai, Jianmin Zheng, and Jiebo Luo

***Abstract*—One weakness in the existing interactive image segmentation algorithms is the lack of more intelligent ways to understand the intention of user inputs. In this paper, we advocate the use of multiple intuitive user inputs to better reflect a user's intention. In particular, we propose a constrained random walks algorithm that facilitates the use of three types of user inputs: (1) foreground and background seed input, (2) soft constraint input, and (3) hard constraint input, as well as their combinations. The foreground and background seed input allows a user to draw strokes to specify foreground and background seeds. The soft constraint input allows a user to draw strokes to indicate the region that the boundary should pass through. The hard constraint input allows a user to specify the pixels that the boundary must align with. Our proposed method supports all three types of user inputs in one coherent computational framework consisting of a constrained random walks and a local editing algorithm, which allows more precise contour refinement. Experimental results on two benchmark data sets show that the proposed framework is highly effective and can quickly and accurately segment a wide variety of natural images with ease.**

***Index Terms*—Interactive image segmentation, digital image editing, multiple user inputs, random walks algorithm.**

I. Introduction

Interactive image segmentation involves minimal user interaction to incorporate user intention into the segmentation process and is an active research area in recent years because it can achieve satisfactory segmentation results that are unattainable by the state-of-the-art automatic image segmentation algorithms. This paper considers the same problem of how to interactively segment a foreground object out from its surrounding background. Our goal is to develop intuitive and intelligent image segmentation algorithms and tools that allow users to interactively guide the segmentation algorithm via a small amount of intuitive interactions until a satisfactory segmentation result that reflects both user intentions and photometric features is achieved.

For a good interactive image segmentation algorithm, there are two basic requirements: (1) given a certain user input, the algorithm should produce intuitive segmentation that reflects the user intent; (2) the algorithm must be efficient so that it can provide instant visual feedback.

W. Yang, J. Cai and J. Zheng are with School of Computer Engineering, Nanyang Technological University, Singapore e-mail: {wxyang, asjfcai, asjmzheng}@ntu.edu.sg

J. Luo is with Eastman Kodak Company, email: jiebo.luo@kodak.com

Note that due to the uploading constraints, the figures included in this PDF file are not of high quality. The high-quality figures are available at http://www.ntu.edu.sg/home/asjfcai/TIP-05173-2009-figures.pdf.

A. Related Work

In general, interactive image segmentation can be classified into two categories: hard segmentation and soft segmentation. Hard segmentation algorithms such as [1], [2] produce a binary map, i.e., a pixel belongs to either foreground or background, while soft segmentation algorithms such as [3], [4] extract a fractional (fuzzy) matte for an image. In this research, we only consider the hard segmentation problem. In the following, we give a brief review on the related interactive image segmentation algorithms and tools.

Early interactive image segmentation algorithms utilize either regional properties such as Adobe's magic wand or boundary properties such as active contour [5] and intelligent scissors [6], [7]. The magic wand tool starts with a small user-specified region. The region grows through connecting neighboring pixels that fall within some adjustable tolerance range of the color statistics of the specified region. With the active contour method, the user is typically asked to place a contour near the desired boundary and the algorithm evolves the boundary to snap to the object contour. The main problem with the active contour method is that the contour is likely to be trapped in a local minimum. The intelligent scissors algorithm requires the user to place points along the desired contour of the foreground object. Dijkstra's shortest path algorithm is used to compute the path between neighboring points. However, in the cases of low contrast or noisy boundaries, the shortest path may "shortcut" the desired boundary. This can be improved by using more effective arc weights [8]. Another problem with the intelligent scissors is that for highly textured (or un-textured) regions there exist many alternative "minimal" paths, which requires a large number of user interactions in order to obtain a satisfactory result.

Most recent algorithms such as the graph cut based methods [1], [9], [10] consider both regional and boundary properties. With the graph cut based methods, an image is modelled as a graph where each node represents a pixel and two neighboring nodes are connected with a weighted edge defined as the distance between the pixel values. Moreover, the graph cut algorithm [1] models foreground and background pixel values according to histograms. Particularly, two virtual nodes are added to the graph to denote the foreground and the background models. The max-flow/min-cut algorithm is employed to classify the nodes. To initialize the models, both foreground and background seeds are needed. The LazySnapping work [9] integrates intuitive user interfaces, including foreground/background strokes and boundary polygon editing, with the graph cut algorithm for easy interactive image seg-

mentation.

The GrabCut method [10] extends the graph cut framework to segment color images, where the foreground and background colors are modeled by Gaussian mixture models. The GrabCut method supports various types of user inputs, including a bounding box to enclose the foreground object, a lasso input for difficult images, foreground and background strokes for local editing, and a boundary brush for matting. The GrabCut method can achieve good performance in segmenting the images whose foreground and background colors are well separable, but its performance is often unsatisfactory for the images whose foreground and background share similar color distributions such as in cluttered or camouflaged images.

One inherent limit for graph cut based methods [1], [9], [10] lies in its underlying assumption that an object's shape is best described by the shape with smallest boundary length. This does not hold for very sophisticated shapes such as bush branches or hair. Such methods do not cope well with noise or interlace effects in videos due to the assumption.

Another popular interactive image segmentation approach is the SIOX algorithm [11], [12] derived from color signature, which has been implemented in image processing softwares including GIMP, Inkscape, and Blender. The SIOX algorithm also depends heavily on the foreground and background color distributions. In addition, it requires a wise selection of representative foreground. The SIOX algorithm works well with noise and videos, but sometimes overcomplicates a shape, introducing holes and other artifacts.

Recently, the random walks algorithm and its extensions [2], [13], which also model an image as a graph, have been adopted for various image processing tasks. It has been demonstrated in [2] that the random walks algorithm can achieve better image segmentation performance than the graph cut algorithm. Similarly, the random walks algorithm requires the input of foreground and background seeds. However, the random walks algorithm lacks a global color distribution model, therefore it is very sensitive to the positions and quantities of foreground and background seeds. The random walks algorithm is essentially an approach that minimizes a Dirichlet energy with boundary conditions, where different boundary conditions (different input seeds) always result in different harmonic functions.

More recently, Bai and Sapiro [14] proposed a geodesic framework based on computing weighted geodesic distances from individual pixels to the user-provided scribbles for interactive image and video segmentation. This algorithm, again, depends on sufficiently separable color distributions of the foreground and the background. The GeoS algorithm [15] further extends the geodesic framework on improving the processing speed and relaxing the connectivity requirement, i.e., each segmented region needs to be connected to the corresponding input stroke. The performance of GeoS is very close to the min-cut algorithm while the computational time is reduced significantly. In fact, as described in [16], the graph cuts, random walks, and geodesics algorithms can be unified under the same optimization framework with different parameter values. The major limitation of the geodesics algorithms is that it is very sensitive to the seed locations since different seed locations result in different geodesic distances for each pixel.

Another class of algorithms [17] uses the image foresting transform (IFT) to provide a common framework for interactive boundary-based segmentation such as [18], [7] and interactive region-based segmentation such as different types of watersheds and geodesic distances [14]. In particular, [19] presents a differential IFT for watershed-based and fuzzy-connected segmentation, whose response time for interactive segmentation corrections is sublinear in practice. The theoretical comparison between the IFT based segmentation methods and the graph cut algorithm is given in [20], which shows that the two methods are closely related, and they produce exactly the same segmentation result under necessary conditions.

B. Our Work

One weakness in the existing interactive image segmentation algorithms is the lack of more intelligent ways to understand the intention of user inputs. Sometimes the user intends to provide cues for regions while at other times the user intends to focus on the boundaries. In other words, a fundamental question we want to ask is: *Can an interactive image segmentation algorithm be intelligent enough to understand the user's intention in different scenarios?* When segmenting a difficult image such as a cluttered or camouflaged scene, the user frequently struggles with laborious local editing because the user cannot effectively communicate his intentions with the underlying segmentation engine. Even with more and more scribbles, the segmentation result still may not be what the user wants.

In this paper, we do not attempt to figure out a single universal intelligent means to acquire user feedback and instead we advocate the use of multiple types of intuitive inputs to better reflect the user's intention under different scenarios. In particular, we propose a constrained random walks algorithm that facilitates the use of three types of user inputs: (1) foreground and background seed input, (2) soft constraint input, and (3) hard constraint input, as well as their combinations. The foreground and background seed input allows a user to draw strokes to specify foreground and background seeds. The soft constraint input allows a user to draw strokes to indicate the region that the boundary should pass through. The hard constraint input allows a user to specify the pixels that the boundary must align with. Note that although the GrabCut [10] and the LazySnapping [9] approaches also support multiple user inputs, different inputs are used at different processing stages. In comparison, the three types of user inputs and any of their combinations can be supported by our coherent computational framework consisting of the constrained random walks algorithm and a local editing algorithm that imposes soft and hard constraints and allows more precise contour refinements.

II. Constrained Random Walks

In this section, we extend the random walks algorithm [2] to a *constrained* random walks algorithm to facilitate the use of various user inputs in interactive image segmentation.

A. The Proposed Model

Similar to the random walks algorithm [2], we formulate the segmentation problem on a graph, where each node represents a pixel and neighboring nodes are connected with undirected edges. In particular, a graph is represented by its vertices and edges as $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$, where $\mathcal{V} = \{v_i\}$ is a set of vertices v_i and $\mathcal{E} = \{e_{ij}\}$ is a set of edges e_{ij} bounded by vertices v_i and v_j. The weight for edge e_{ij} is denoted as w_{ij}, and the degree of node v_i is defined as $d_i = \sum_j w_{ij}$, i.e., the sum of the weights of all the edges that incident on v_i.

In the random walks algorithm, the user input includes foreground seeds S_F and background seeds S_B, where $S_F \subset \mathcal{V}$, $S_B \subset \mathcal{V}$, and $S_F \bigcap S_B = \emptyset$. Let p_i $(0 \leq p_i \leq 1)$ denote the probability that a random walker starting from v_i will first arrive at one of the foreground seeds before reaching any of the background seeds. Clearly, $p_i = 1$ for any $v_i \in S_F$ and $p_i = 0$ for any $v_i \in S_B$. For any of the remaining vertices $v_i \in \mathcal{V} \setminus (S_F \cup S_B)$, the random walks algorithm suggests

$$p_i = \frac{1}{d_i} \sum_{e_{ij} \in \mathcal{E}} w_{ij} \cdot p_j \tag{1}$$

This leads to a linear system of equations with p_i for $v_i \in \mathcal{V} \setminus (S_F \cup S_B)$ as unknowns. Solving the equations gives the probability of vertex v_i first arriving at S_F. Finally, the foreground object is segmented as the set of pixels whose probabilities are not less than 1/2.

We now incorporate two other types of user inputs as constraints into the random walks algorithm. We call such an extension as *constrained random walks*. In particular, boundary brush strokes that roughly mark parts of the boundary are introduced as the *soft* constraint. A vertex on which the soft constraint is imposed has the property that the difference between its probability and 1/2 is within a small prescribed range $[-\epsilon, \epsilon]$. The second type of user inputs, boundary pixel selector, which selects pixels on the desired contour, is introduced as the *hard* constraint. A vertex on which the hard constraint is imposed has a probability of 1/2.

Let S_S and S_H denote soft boundary seeds and hard boundary seeds, respectively. We define the constrained random walks problem as solving the following equations:

$$p_i = \frac{1}{d_i} \sum_{e_{ij} \in \mathcal{E}} w_{ij} \cdot p_j, \quad v_i \in \mathcal{V} \setminus (S_F \cup S_B \cup S_H) \tag{2}$$
$$s.t., \begin{cases} p_i = 1, & v_i \in S_F \\ p_i = 0, & v_i \in S_B \\ p_i = 0.5, & v_i \in S_H \\ |p_i - 0.5| \leq \epsilon, & v_i \in S_S, \epsilon \approx 0^+ \end{cases}$$

It is difficult to find an effective solver for the above problem due to the soft constraint. Therefore, we reformulate the problem into

$$\min \sum_{e_{ij} \in \mathcal{E}} w_{ij}(p_i - p_j)^2 + \lambda \sum_{v_i \in S_S} (p_i - 0.5)^2 \tag{3}$$
$$s.t., \begin{cases} p_i = 1, & v_i \in S_F \\ p_i = 0, & v_i \in S_B \\ p_i = 0.5, & v_i \in S_H \end{cases}$$

where λ is a tradeoff factor controlling the importance of the difference between the probability of each soft-constraint vertex and 1/2.

Differentiating the objective function of Eq. (3) with respect to each p_i for $v_i \in \mathcal{V} \setminus (S_F \cup S_B \cup S_H)$ and setting it equal to zero, we arrive at

$$p_i = \begin{cases} \frac{1}{d_i + \lambda}(\sum_j w_{ij} \cdot p_j + \lambda \cdot 0.5), & v_i \in S_S \\ \frac{1}{d_i}(\sum_j w_{ij} \cdot p_j), & v_i \in \mathcal{V} \setminus (S_F \cup S_B \cup S_H \cup S_S) \end{cases} \tag{4}$$

This can be considered as adding a virtual neighbor vertex with probability of 1/2 to each soft-constraint vertex through a virtual edge with weight λ. Empirically we set $\lambda = 1$ for all the experiments.

In this way, the constrained random walks problem becomes the problem of solving a linear system of equations shown in Eq. (4). Many efficient methods are available for solving such a sparse linear system. Note that the connectivity property of the random walks algorithm remains true for the proposed constrained random walks.

B. Edge Weights

To achieve a good segmentation, the edge weights play a critical role since each edge weight w_{ij} describes the likelihood of a random walker moving to the neighboring node. Each weight should be defined based on the distance between two neighboring pixels/nodes. In the random walks algorithm [2], the distance between two nodes is defined as $d_{ij} = ||g_i - g_j||$, which is the Euclidean distance in color.

Our studies show that the performance of the random walks algorithm is sensitive to strokes' positions mainly because the random walks algorithm only utilizes the information of color changes without considering absolute color information [13]. Therefore, even in the cases that the foreground and background colors are sufficiently separable, the random walks algorithm may require more carefully drawn strokes compared to the GrabCut. On the other hand, prior models that describe the color/feature distributions are often used in other popular interactive image segmentation algorithms such as the GrabCut algorithm.

In fact, prior models have also been incorporated into the random walks framework in [13], where two virtual nodes are added in the graph to represent the foreground and the background. The foreground/background node is connected to every pixel node and the edge weight is assigned to the probability that the pixel fits the foreground/background prior model. However, the drawback of this work is that the segmentation result often contains multiple disconnected components, which is undesirable for single object segmentation.

In this study, we propose to directly incorporate prior models into the distance function to avoid the above-mentioned disconnection problem. In particular, edge weight w_{ij} and the corresponding distance are defined as

$$w_{ij} = \exp(-\beta \cdot d_{ij}^2)$$
$$d_{ij}^2 = (1 - \alpha)||g_i - g_j||^2 + \alpha(Pr(v_i) - Pr(v_j))^2 \tag{5}$$

where the second term $(Pr(v_i) - Pr(v_j))^2$ is the prior term, $Pr(v_i)$ denotes the normalized probability that node v_i fits the

foreground prior model, $\alpha \in [0,1]$ is a tradeoff factor, and β is a scaling factor. The values of $||g_i - g_j||^2$ and $(Pr(v_i) - Pr(v_j))^2$ are normalized to $[0,1]$ individually.

The foreground and the background are modeled by the *Gaussian mixture model* (GMM) and their seeds S_F and S_B are used to estimate the model parameters. Let $Pr(v_i|\mathcal{F})$ ($Pr(v_i|\mathcal{B})$) denote the probability that node v_i fits the foreground (background) GMM. The normalized probability $Pr(v_i)$ is defined as

$$Pr(v_i) = \frac{-\log Pr(v_i|\mathcal{F})}{-\log Pr(v_i|\mathcal{F}) - \log Pr(v_i|\mathcal{B})} \quad (6)$$

According to Eq. (5), it is clear that the second term $(Pr(v_i) - Pr(v_j))^2$ should dominate when the foreground and background colors are well separable. Otherwise, the first term $||g_i - g_j||^2$ should dominate. This gives us a hint on how to select the parameter α. In our experiments, we set α to be the distance between the foreground and background GMMs. A natural measure between two distributions is the Kullback-Leibler divergence. We use the Monte-Carlo simulations to approximate the KL-divergence between $\mathcal{F}$ and $\mathcal{B}$. More specifically, we define

$$\alpha = \frac{1}{n}\sum_{i=1}^{n} \left|\frac{\log Pr(v_i|\mathcal{F}) - \log Pr(v_i|\mathcal{B})}{\log Pr(v_i|\mathcal{F}) + \log Pr(v_i|\mathcal{B})}\right| \quad (7)$$

where n is the total number of pixels. After determining the parameter α, we can calculate all the distances. The only free parameter left in the proposed algorithm is β, which is empirically set to 3 for all the experiments.

We would like to point out that the proposed edge weighting method could be applied to any graph-based algorithm such as random walks, graph cuts, and normalized cuts [21]. Additionally, a similar idea was given in [22], where Parzen windows were used to estimate a foreground/background distribution for setting edge weights in 3D image segmentation.

III. Local Contour Deformation

Through the proposed constrained random walks algorithm, a continuously valued probability map is computed where the value of each pixel indicates its probability of belonging to the foreground. An initial contour $\mathcal{C}$ is obtained by thresholding the probability map at 1/2. As will be shown, the above algorithm can still have problems for segmenting along weak boundaries. One way to improve the segmentation performance is to introduce more user inputs and then re-run the algorithm or formulate the segmentation editing task as another global energy minimization problem such as [23]. However, this may often affect the final segmentation results globally. Unexpected fluctuation effect may occur during the process of interactive object cutout [24] and re-running the algorithm or solving another global optimization increases the system complexity. On the other hand, in many circumstances, there is no need to cause a global change while only local editing is desired.

Therefore, we further propose a local refinement step, where only additional hard and soft boundary constraints are allowed to be used to indicate the problematic boundaries. Even without additional hard and soft boundary constraints, this local refinement still needs to be performed if there is any soft or hard boundary stroke being input in the previous global stage. This is because the constrained random walks algorithm itself cannot guarantee achieving the purposes of both hard and soft boundary constraints. For example, it can happen that all the neighboring nodes of a hard constraint node have probabilities larger than 1/2 or less than 1/2, for which the resulting contour will not pass through the hard constraint node. Similarly, for a soft boundary stroke, if the weighting parameter λ is too small, the resulting contour may not pass through the stroke. On the other hand, if λ is too large, the pixels masked by the soft boundary strokes will have probability values very close to 1/2, for which the precise boundary is difficult to locate by thresholding.

For the local refinement step, our basic idea is to first determine the pixels/positions that the contour must pass and then build the correspondences between these pixels and the pixels on the initial contour. After that, the initial contour is deformed with the correspondences used as positional constraints and the rest of the pixels on the initial contour as stay-put constraints. In this way, the contour can be pulled to the specified boundary locations locally, and the smoothness near the pulled positions can be preserved.

A. Optimal Path in Soft Boundary Stroke

Within each soft boundary stroke, an optimal path called soft boundary path needs to be searched first. We use Dijkstra's shortest path algorithm to find the path. The edge weight is adapted from the "live-wire" path selection tool in [18], [7]. In particular, Let v_p and v_q denote two neighboring pixels. The local cost e_{pq} of the directed edge from v_p to v_q is defined as a weighted sum of three components: Laplacian zero crossings f_z, gradient magnitude f_g, and the gradient directional cost $f_d(p,q)$, i.e.,

$$e_{pq} = \{0.1 \cdot f_z(q) + 0.6 \cdot f_g(q) + 0.3 \cdot f_d(p,q)\}/len, \quad (8)$$

where the division by len, denoting the length of the path from v_p to v_q, is to avoid the "shortcut" problem. The weights in Eq. (8) are empirical values.

The purpose of having the Laplacian zero-crossing term in Eq. (8) is for edge localization. The gradient magnitude term is to distinguish between a strong edge and a weak edge. The gradient direction or orientation term adds a smoothness constraint to the boundary by associating a relatively high cost for sharp changes. The details about how to compute f_z, f_g, and f_d can be found in [18]. Note that the proposed local cost in Eq. (8) is almost the same as that in [18] except that we use different weights and apply the normalization by len.

B. Finding Correspondences

Correspondences between the user input soft and hard boundary points and the initial contour $\mathcal{C}$ obtained in the global stage are established before contour deformation. The correspondence for a hard constraint point is defined as the point on the contour with minimum geometric distance. For soft boundary constraints, we match the points from the initial

contour to the soft boundary path inside the stroke. First, the two end points of the soft boundary path are matched to the points on the contour with minimum geometric distance, and then the in-between points of the soft boundary path and the contour are matched by bilinear interpolation. We obtain a correspondence set as

$$\begin{aligned} \{\mathbf{v}_1^h, \mathbf{v}_2^h, \cdots, \mathbf{v}_{k1}^h\} &\rightarrow \{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{k1}\} \\ \{\mathbf{v}_1^s, \mathbf{v}_2^s, \cdots, \mathbf{v}_{k2}^s\} &\rightarrow \{\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_{k2}\} \end{aligned} \tag{9}$$

where $\mathbf{h}_i$ and $\mathbf{s}_i$ are the hard and soft boundary constraint points, and $\mathbf{v}_i^h$ and $\mathbf{v}_i^s$ are the corresponding points on the initial contour.

C. Refinement by Contour Deformation

The contour deformation problem can be summarized as: given the initial contour $\mathcal{C}$, the positional constraints resulted from the correspondences for the hard and soft boundary points (Eq. (10)), and the stay-put constraints using the rest of the pixels in the initial contour, find the new contour with shape preservation and smoothness. We formulate this problem as

$$\begin{aligned} &\arg\min_{\mathbf{v}'}(||L\mathbf{v}' - L\mathbf{v}||^2 + \omega \sum_j ||\mathbf{v}_j^{s\prime} - \mathbf{s}_j||^2) \\ &+ \sum_{j \in \mathcal{C}_{stay}} f(dist(j))||\mathbf{v}_j' - \mathbf{v}_j||^2, \quad s.t., \mathbf{v}_j^{h'} = \mathbf{h}_j \end{aligned} \tag{10}$$

where ω (empirically set to 1 in our experiments) is a tradeoff parameter for soft boundary constraint points, and $\mathbf{v}$ and $\mathbf{v}'$ denote two column vectors whose elements $\mathbf{v}_j$ and $\mathbf{v}'_j$ are the vertices on the initial contour $\mathcal{C}$ and the deformed contour $\mathcal{C}'$, respectively. As the complexity of contour deformation is very low, we use all the pixels on the contour for deformation. It is also possible to set a minimum distance between the neighboring sampling points on the contour.

The first term $||L\mathbf{v}' - L\mathbf{v}||^2$ in Eq. (10) is adapted from the Laplacian mesh processing [25] to preserve the global shape with the transform matrix L defined as

$$L = I - D^{-1}A, \tag{11}$$

where D is a matrix with $D_{ii} = 2$ and A is the adjacent matrix defined as

$$A_{ij} = \begin{cases} 1 & |i-j| = 1 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

For the stay-put constraints in Eq. (10), we use the distance function $f(dist(i))$ as the weight. Particularly, $dist(i)$ is the normalized geometric distance from pixel $v(i)$ to the input strokes and the function f can be any monotonic increasing function such that pixels near the input strokes have small weights and those far away from the input strokes have large weights. It follows that the contour pixels far away from the inputted stroke that are considered to be of high confidence are unlikely to be moved. In this research, we use $f(x) = x$. In addition, the pixels v_i with $f(dist(i))$ larger than a predefined threshold are excluded from the contour deformation step.

The minimization problem (10) can be converted into the form of a sparse linear system by differentiating the objective function with respect to the unknown vertices and then letting these partial derivatives equal zero. Solving the sparse linear system thus gives the new contour.

IV. Experiments and Discussions

In this section, experiments are conducted to evaluate the effectiveness of the proposed framework. The test images come from the Berkeley segmentation dataset[1] and the MSRC ground truth dataset provided in [26][2].

A. With Only Foreground and Background Strokes

We first consider the case that uses only foreground and background strokes as inputs. In this case, our proposed framework degenerates to the random walks algorithm [2] except that our approach incorporates the prior information into the edge weights as shown in Eq. (5).

Fig. 1 shows the segmentation results of three test images using the random walks algorithm and our approach. Note that for fair comparison, the same foreground and background strokes are used to initiate both algorithms. It can be seen that the proposed method significantly outperforms the random walks algorithm on the three test images whose foreground and background colors are well separable, which demonstrates the effectiveness of the newly incorporated edge weights.

We have also tested the proposed algorithm on the MSRC ground truth data set [26], which consists of 50 test images. To the best of our knowledge, this is currently the only publicly available image segmentation data set that provides trimaps and ground truth. Table I summarizes the achieved error rates by the proposed algorithm and other state-of-the-art algorithms. The error rate is defined as

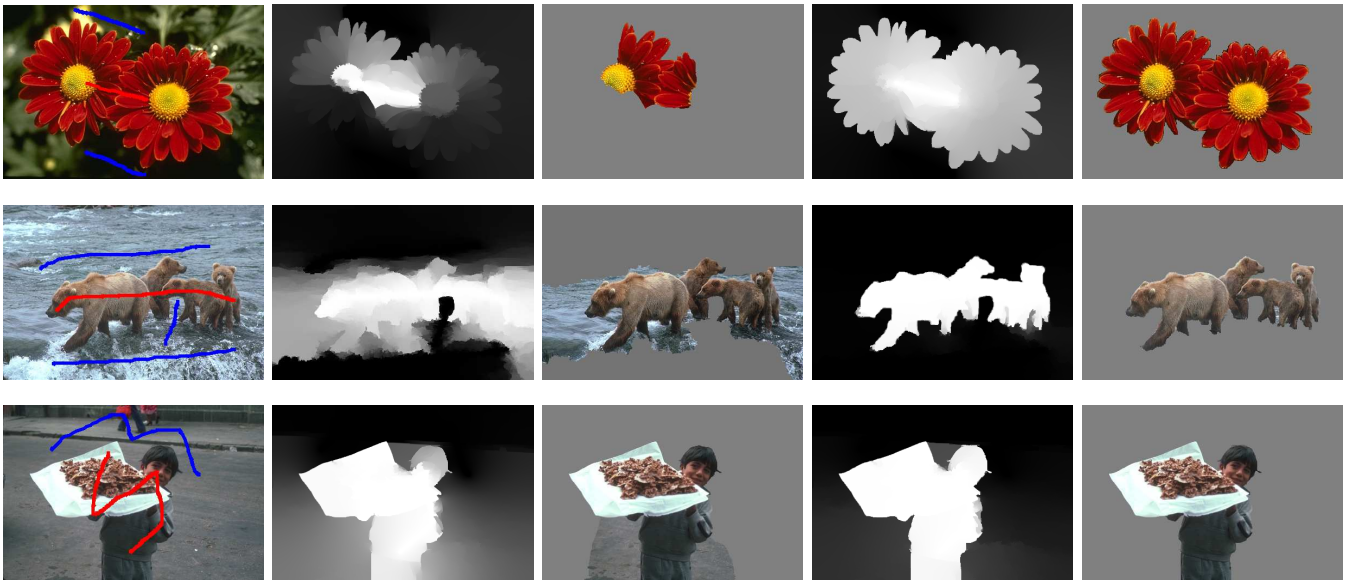
$$\epsilon = \frac{\text{no. misclassified pixels}}{\text{no. pixels in unclassified region}}, \tag{13}$$

where "misclassified pixels" excludes those from the unclassified region of the expert trimap [11], [26]. Our proposed algorithm achieves an error rate of 4.08% with a variance of 3.72%. For fair comparison, all the algorithms use exactly the same trimaps provided with the MSRC data set as the user inputs. The error rates for other state-of-the-art algorithms are either directly quoted from the best results reported in literature or obtained through our implementation. The LazySnapping code is obtained online from ETH Zurich (http://www.cg.inf.ethz.ch/teaching/former/imagesynthesis_06/miniprojects/p2/index.html).

Note that a simple adaptive threshold method was reported in [27] that can be combined with the existing algorithms such as the random walks to further reduce the error rate. However, the adaptive threshold method is very specific to the MSRC data set or a boundary brush tool where the unseeded region only covers a small band along the object boundary. It is not a general method that can be applied to any segmentation. Specifically, the adaptive threshold method does not work well

[1] Available at http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

[2] Available at http://research.microsoft.com/en-us/um/cambridge/projects/visionimagevideoediting/segmentation/grabcut.htm



(a) input image (b) RW probability map (c) RW foreground (d) our probability map (e) our foreground

Fig. 1. Comparison between the random walks algorithm and the global step of the proposed framework to demonstrate the effectiveness of the proposed edge weights. From left to right: (a) input images with strokes (red for foreground and blue for background); (b) the probability maps by random walks; (c) the segmentation results by random walks; (d) the probability maps by the proposed method (showing more confident separation); (e) the segmentation results by the proposed method. The KL-divergence values of the three test images are 0.78, 0.84 and 0.53, respectively. The parameter β in Eq. (5) is fixed to 3 and 500 for our method and the random walks algorithm, respectively. Best viewed electronically (with zoom-in).

for images with large unknown area. Therefore, we do not compare with the results with the adaptive threshold method reported in [27].

For most of the images in the MSRC data set, the proposed method achieves very low error rates. High error rates occur in images where the input foreground seeds only cover a small portion of the foreground and thus do not cover all the distinct colors of the foreground, while pixels with similar colors are masked by the background seeds. For such cases, the GrabCut algorithm and the random walks algorithm also perform poorly (and usually worse).

Moreover, by simply drawing one or two additional foreground and/or background strokes, or by the aid of soft and/or hard constraints, the error rates of such failure cases can be significantly further reduced using our unified approach. One example is that by adding several foreground and background strokes to the seven images (with an error rate higher than 9% by our method initially) in the MSRC data set, we reduce the initial errors dramatically and the overall average error rate of the 50 images is dropped from 4.08% to 2.84%.

We would like to point out that, although both our method and LazySnapping use multiple types of user inputs, our method is quite different from LazySnapping. In particular, in terms of user interface, LazySnapping uses different algorithms to handle different inputs. To the user, the region segmentation and boundary editing are two separate steps. On the contrary, our work supports multiple intuitive inputs and any of their combinations under one computational framework. In terms of speed, our approach is faster as LazySnapping refines the entire contour in the boundary editing step using energy minimization by graph cut, while in our framework boundary editing is a local deformation process.

TABLE I
ERROR RATE COMPARISON USING THE MSRC DATASET WITH EXACTLY THE PROVIDED TRIMAPS.

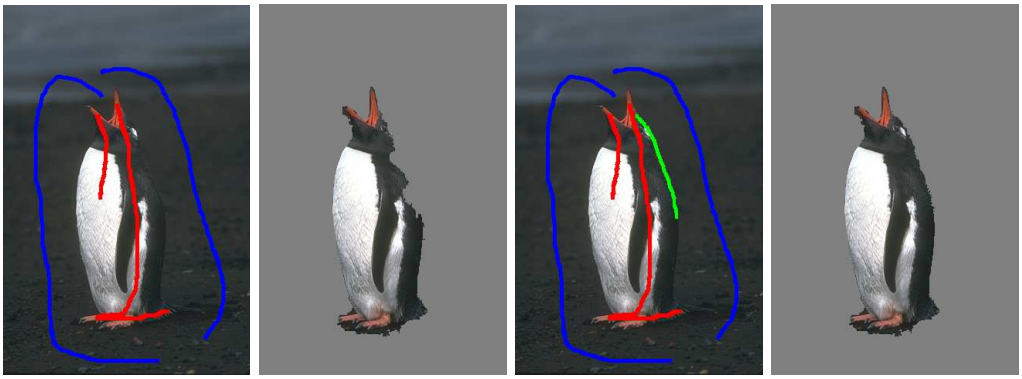
Method	Error rate
GMMRF [26]	7.9% (reported in [26])
GrabCut [10]	5.66% (our implementation)
SIOX [11]	9.1% (reported in [11])
Random walks [2]	5.4% (reported in [27])
LazySnapping [9]	6.65%
Proposed	4.08%

B. With Additional Soft and Hard Constraints

Fig. 2 shows an example of utilizing additional constraint in the constrained random walks framework. As illustrated in the figure, by only inputting the foreground and background strokes, the weak boundary between the object and the background cannot be detected precisely. However, with only one additional soft boundary stroke (which is only partially drawn on the boundary), a much better result is obtained.

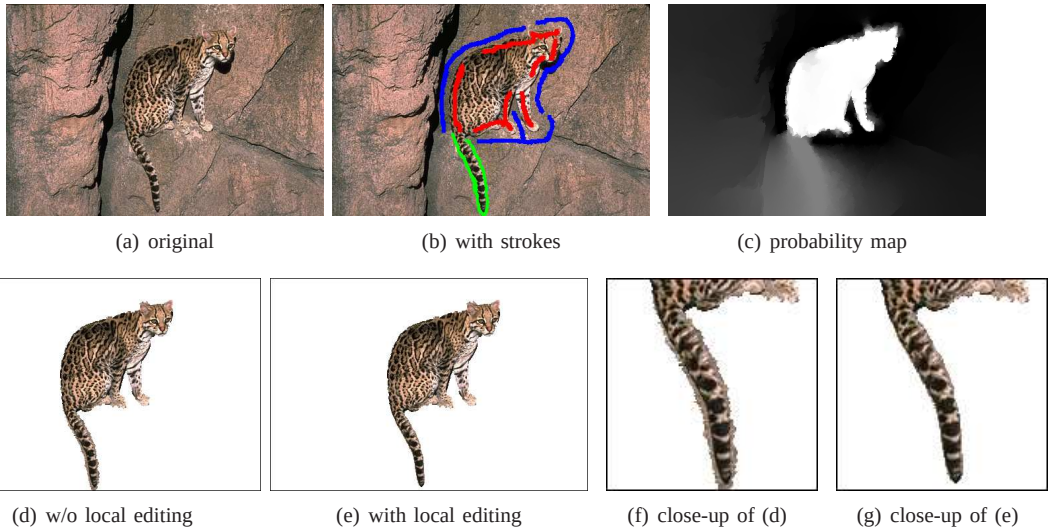
We now evaluate the proposed local editing algorithm. Fig. 3 shows the effect of the local editing with a soft boundary constraint. The computed probability map in Fig. 3(c) tells that the pixels near the soft boundary (the tail of the cat) have probability values very close to 1/2. Thus, the thresholding operation does not lead to a clean boundary within the soft stroke, as shown in Fig. 3(f). On the other hand, the local editing algorithm, i.e., first finding the shortest path within the soft stroke that captures a good boundary and then deforming the initial contour to the shortest path, results in a better object extraction, as shown in Fig. 3(g).

Fig. 4 shows the effect of the local editing with hard boundary constraints. In particular, after the global step with



(a) input image (b) foreground 1 (c) + soft stroke (d) foreground 2

Fig. 2. Comparing the results with and without soft constraint for the constrained random walks algorithm. From left to right: (a) input image with strokes (red for foreground and blue for background); (b) corresponding result; (c) with an additional soft boundary stroke (in green); (d) corresponding new result.



(a) original (b) with strokes (c) probability map

(d) w/o local editing (e) with local editing (f) close-up of (d) (g) close-up of (e)

Fig. 3. Comparing the results with and without local editing. (a) original image; (b) original image with strokes (red, blue and green for foreground, background and soft boundary strokes, respectively); (c) the probability maps by the global step of the proposed framework; (d) the segmentation result without local editing; (e) the segmentation result with local editing; (f) close-up of (d); (g) close-up of (e).

the foreground and background strokes, the object is fairly well segmented except some small inaccuracies as shown in the close-up in Fig. 4(c). This problem can be easily fixed by using hard boundary constraints. Specifically, by marking five hard boundary pixels, the proposed local editing algorithm snaps the initial contour to the specified boundary pixels through local contour deformation, which results in a smoother and more accurate object contour.

C. More Comparisons

Fig. 5 shows the comparison of the segmentation results among the GrabCut algorithm, the random walks algorithm, and the proposed framework. For the case using the GrabCut algorithm, a rectangle covering the object is not sufficient to obtain a good result. Thus, foreground and background strokes need to be continuously added until a reasonable result is achieved. For the case using the random walks algorithm, adding more strokes may cause unexpected fluctuation, i.e., the previously correctly labeled regions change their labels when more strokes are added. On the contrary, by using the proposed framework that facilitates different types of user inputs, we are able to refine the initial results more efficiently and effectively (in two steps interaction as opposed to three).

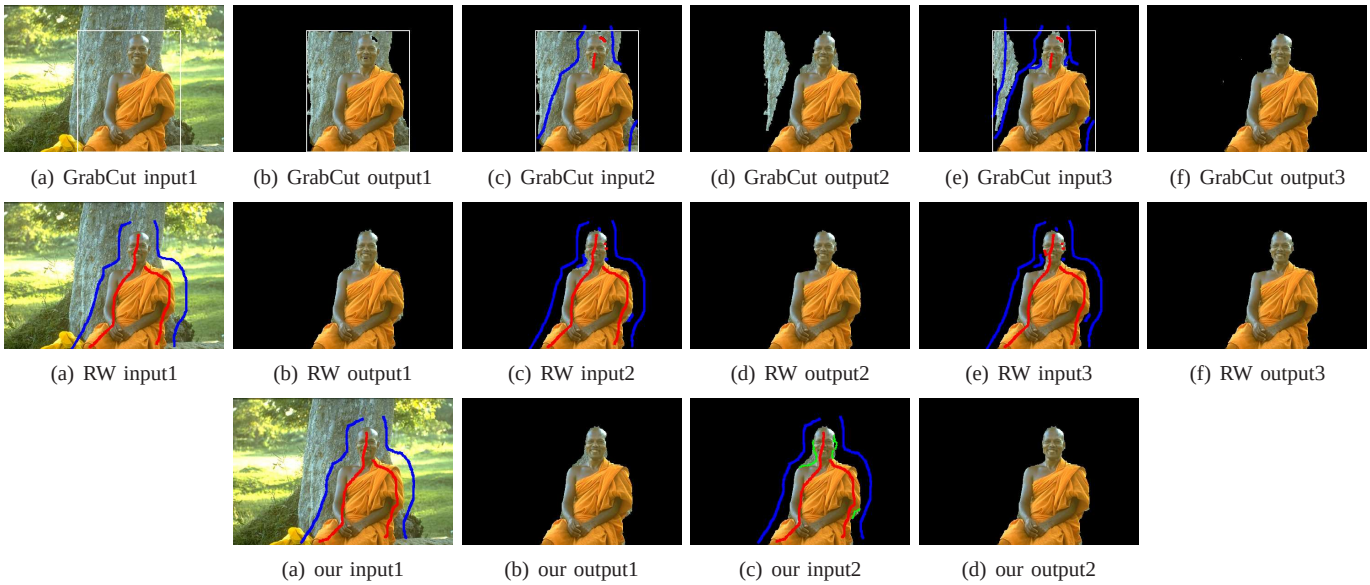
Compared with the popular GrabCut algorithm that requires iterative optimization, the proposed framework only requires to solve sparse linear equations and is thus much faster in speed. The constrained random walks module typically takes less than three seconds to process an image with a resolution of 640×480, and the local editing module can generate the result virtually instantly. For the MSRC data set with the provided trimaps, the average processing time of our proposed framework is 1.48 seconds while that of Grabcut is 4.64 seconds.

D. Limitations

The proposed constrained random walks algorithm follows the random walks framework. As mentioned, without the soft and hard boundary constraints, the only difference between our method and the random walks lies in the edge weighting. Therefore, the performance of the proposed method is close

(a) input image (b) foreground 1 (c) close-up of (b) + hard constraints (d) foreground 2

Fig. 4. An example of the local editing with hard boundary constraints. (a) the input image with strokes (red for foreground and blue for background); (b) the segmentation result by the global step of the proposed framework; (c) close-up of (b) with additional five hard boundary pixels in magenta; (d) the segmentation result after local editing.



(a) GrabCut input1 (b) GrabCut output1 (c) GrabCut input2 (d) GrabCut output2 (e) GrabCut input3 (f) GrabCut output3

(a) RW input1 (b) RW output1 (c) RW input2 (d) RW output2 (e) RW input3 (f) RW output3

(a) our input1 (b) our output1 (c) our input2 (d) our output2

Fig. 5. Comparing the segmentation results of the GrabCut algorithm, the random walks algorithm, and the proposed framework. From top to bottom: the results of GrabCut, the results of random walks (RW), and the results of the proposed framework. The red, blue, and green strokes denote foreground, background, and soft boundary strokes, respectively. Best viewed electronically (with zoom-in).

to random walks except for images whose foreground and background colors are well separable, where the proposed method outperforms the random walks. With the soft and hard boundary tools as a whole, the proposed framework provides an easier tool for users to segment difficult images, such as noisy images and images with complex shapes.

In general, the proposed framework performs well on most images with straightforward user inputs. It can do well for camouflaged images and images with thin structures if the user inputs are carefully placed. One example is shown in Fig. 3, where foreground and background strokes are carefully placed at each side of the object boundary in order to well segment the object.

With a certain amount of user interaction, failure cases can occur for hairy objects or thin structures whose color overlapped with the background color. In this case, the proposed edge weighting degenerates to that of the random walks algorithm, while intensive additional user efforts are required to delineate the fine boundary using the proposed boundary tools. We would like to point out that since the proposed framework is a hard segmentation method, it cannot well handle transparent or semi-transparent boundaries such as semi-lucent hair, for which we have to rely on matting techniques.

V. Conclusions

In this paper, we have proposed an interactive image segmentation framework that consists of two components: constrained random walks and local contour deformation. The proposed framework supports multiple intuitive types of user inputs and therefore combines the advantages of different user interactions. The foreground and background brushes are the most commonly used interaction tools as they are easy to use and instructive to the algorithms. The soft boundary brush and the hard boundary pixel selector are extremely useful to handle weak boundaries, where adding more foreground or background strokes may cause unexpected fluctuation in the segmentation results. These tools enable the proposed framework to work fast and accurately with ease. The superior performance of the algorithm has been demonstrated by a number of experiments on the benchmark data sets.

The contributions of this paper can be summarized as follows. First, the proposed constrained random walks algorithm together with the proposed local editing algorithm supports

the three types of user inputs and their combinations in a coherent and unified framework. Second, the region prior term is included in the edge weights so that the proposed constrained random walks algorithm does not lose the connectivity property and is less demanding on the positions and quantities of the user input strokes than the original random walks algorithm [2]. Third, the proposed local editing algorithm also allows additional local refinement to reach a satisfactory segmentation.

The proposed framework can be extended in several ways. First, its runtime can be further improved. In particular, similar to the random walks algorithm, the complexity of the proposed method lies in solving a sparse linear system, whose dimension depends on the number of unknown pixels in the image and the adjacency structure (e.g., 4-connected or 8-connected). There are many efficient algorithms for solving sparse linear equations and we adopt the sparse direct linear solver implemented in TAUCS (http://www.tau.ac.il/~stoledo/taucs/). Without code optimization, the response time of the proposed framework is typically around 5 seconds for a 640×480 image on a PC with Intel 2.67GHz CPU and 2GB RAM. The response time can be greatly improved through the graphics processing unit (GPU) implementation [28]. The local editing step also requires solving of a sparse linear system, whose dimension depends on the sampling of the object contour. As the number of pixels on the object contour is much smaller than image size, even without sampling, the local editing step gives immediate response.

Second, it is interesting to extend the proposed method to soft segmentation and video segmentation. In fact, the random walks algorithm has been successfully extended for soft segmentation [28]. Similarly, we could integrate effective boundary matting tools into the proposed method for soft segmentation. Extending our method to video segmentation is not straightforward. This is because video data consists of thousands of video frames. It is impossible for a user to provide inputs for each video frame. Thus, it becomes critical to design an intuitive and user-friendly interface to efficiently acquire user feedback.

Last but not least, it is meaningful to conduct a user study to compare our method with different interactive image segmentation algorithms in terms of usability. The user study should involve many professional and unprofessional users to segment a large number of test images. With the intensive user study, it is possible to create a metric to measure the amount and the complexity of interaction that is required for an interactive image segmentation algorithm to achieve satisfactory results.

REFERENCES

[1] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images," in *ICCV*, 2001, pp. 105–112.

[2] L. Grady, "Random walks for image segmentation," *IEEE Trans. PAMI*, vol. 28, no. 11, pp. 1768–1783, Nov. 2006.

[3] A. Levin, D. Lischinski, and Y. Weiss, "A closed form solution to natural image matting," in *CVPR*, 2006, pp. 61–68.

[4] A. Levin, A. Rav-Acha, and D. Lischinski, "Spectral matting," in *CVPR*, 2007, pp. 1–8.

[5] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *IJCV*, pp. 321–331, 1988.

[6] E. N. Mortensen and W. A. Barrett, "Interactive segmentation with intelligent scissors," *Graphical Models in Image Processing*, vol. 60, no. 5, pp. 349–384, 1998.

[7] A. Falcão, J. Udupa, and F. Miyazawa, "An ultra-fast user-steered image segmentation paradigm: Live-wire-on-the-fly," *IEEE Trans. on Medical Imaging*, vol. 19, no. 1, pp. 55–62, Jan 2000.

[8] P. Miranda, A. Falcão, and J. Udupa, "Synergistic arc-weight estimation for interactive image segmentation using graphs," *Computer Vision and Image Understanding*, vol. 114, no. 1, pp. 85–99, Jan. 2010.

[9] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum, "Lazy snapping," in *ACM Siggraph*, 2004, pp. 303–308.

[10] C. Rother, V. Kolmogorov, and A. Blake, ""GrabCut": Interactive foreground extraction using iterated graph cuts," in *ACM Siggraph*, 2004, pp. 309–314.

[11] G. Friedland, K. Jantz, and R. Rojas, "SIOX: Simple interactive object extraction in still images," in *IEEE International Symposium on Multimedia (ISM'05)*, Dec. 2005, pp. 253–259.

[12] G. Friedland, K. Jantz, T. Lenz, F. Wiesel, and R. Rojas, "A practical approach to boundary accurate multi-object extraction from still images and videos," in *IEEE International Symposium on Multimedia (ISM'06)*, Dec. 2006, pp. 307–316.

[13] L. Grady, "Multilabel random walker image segmentation using prior models," in *CVPR*, 2005, pp. 763–770.

[14] X. Bai and G. Sapiro, "A geodesic framework for fast interactive image and video segmentation and matting," in *ICCV*, 2007, pp. 1–8.

[15] A. Criminisi, T. Sharp, and A. Blake, "GeoS: Geodesic image segmentation," in *ECCV*, 2008, pp. 99–112.

[16] A. K. Sinop and L. Grady, "A seeded image segmentation framework unifying graph cuts and random walks which yields a new algorithm," in *ICCV*, 2007, pp. 1–8.

[17] A. Falcão, J. Stolfi, and R. Lotufo, "The image foresting transform: Theory, algorithms, and applications," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 19–29, 2004.

[18] E. N. Mortensen and W. A. Barrett, "Intelligent scissors for image composition," in *Computer Graphics and Interactive Techniques*, 1995, pp. 191–198.

[19] A. X. Falcão and F. P. G. Bergo, "Interactive volume segmentation with differential image foresting transforms," *IEEE Trans. on Medical Imaging*, vol. 23, no. 9, pp. 1100–1108, 2004.

[20] P. A. V. Miranda and A. X. Falcão, "Links between image segmentation based on optimum-path forest and minimum cut in graph," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 128–142, 2009.

[21] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. PAMI*, vol. 22, no. 8, pp. 888–905, August 2000.

[22] L. Grady and M.-P. Jolly, "Weights and topology: A study of the effects of graph construction on 3D image segmentation," in *MICCAI*, vol. 1, 2008, pp. 153–161.

[23] L. Grady and G. Funka-Lea, "An energy minimization approach to the data driven editing of presegmented images/volumes," in *MICCAI*, vol. 2, 2006, pp. 888–895.

[24] C. Wang, Q. Yang, M. Chen, X. Tang, and Z. Yu, "Progressive cut," in *Proc. ACM Multimedia*, 2006, pp. 251–260.

[25] O. Sorkine, "Laplacian mesh processing," in *State of The Art Report, Eurographics*, 2005, pp. 53–70.

[26] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr, "Interactive image segmentation using an adaptive GMMRF model," in *ECCV*, 2004, pp. 428–441.

[27] O. Duchenne and J.-Y. Audibert, "Segmentation by transduction," in *CVPR*, 2008, pp. 1–8.

[28] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann, "Random walks for interactive alpha-matting," in *Proc. VIIP*, 2005, pp. 423–429.