Noname manuscript No.
(will be inserted by the editor)

An Improved Image Graph for Semi-Automatic Segmentation

Daniel Freedman

Received: date / Accepted: date

Abstract The problem of semi-automatic segmentation has attracted much interest over the last few years. The Random Walker algorithm [1] has proven to be quite a popular solution to this problem, as it is able to deal with several components, and models the image using a convenient graph structure. We propose two improvements to the image graph used by the Random Walker method. First, we propose a new way of computing the edge weights. Traditionally, such weights are based on the similarity between two neighbouring pixels, using their grayscale intensities or colours. We substitute a new definition of weights based on the *probability distributions* of colours. This definition is much more robust than traditional measures, as it allows for textured objects, and objects that are composed of multiple perceptual components. Second, the traditional graph has a vertex set which is the set of pixels, and edges between each pair of neighbouring pixels. We substitute a smaller, irregular graph based on Mean Shift oversegmentation. This new graph is typically several orders of magnitude smaller than the original image graph, which can lead to a major savings in computing time. We show results demonstrating the substantial improvement achieved when using the proposed image graph.

Keywords image segmentation · graph · probability distribution · random walk

1 Introduction

The problem of semi-automatic or interactive segmentation has attracted quite a bit of interest from the computer vision, image processing, and computer graphics communities over the last number of years. The general idea is to segment an image into two or more separate regions, each corresponding to a particular object (or the background), with the aid of some user input. The user input can take on a variety of forms. A common scenario has the user "scribbling" on each of the objects of interest, with each scribble corresponding to a given object, and indicating a small number of pixels that are contained within that object; see Figure 1. Other types of user inputs, such as bounding boxes [2] and the like, are possible as well. In any case, the user input should be simple enough that a user can provide the input in a few seconds; for example, a careful delineation of an object's boundary is deemed far too effort-intensive. Given this minimal user input, the goal is to segment the image into objects, as shown in Figure 1.

Semi-automatic segmentation is an attractive problem from two perspectives. First, there are several applications which rely on this technology. In the arena of image-editing tools, useful in the graphic arts and elsewhere, extraction of objects from images is an indispensable tool; and the more minimal the user input, the faster the graphic artist is able to complete the required task. In the field of medical imaging, it is often important for a physician to quickly and accurately segment a particular organ or tumour; this allows for image-guided radiation therapy planning, amongst many other possibilities.

The second perspective from which semi-automatic segmentation is attractive is the algorithmic perspective. The problem itself is an interesting and challenging one, as one is essentially interesting in propagating information from a small set of known samples – the user input – to the entire image. Thought of in this way, the problem acquires an interesting mathematical flavour.

Of the several algorithms that have been proposed to solve the semi-automatic segmentation problem (see Section 2 for a review), the Random Walker of Grady *et al.* [3, 1, 4] has several inherent advantages. First, unlike most tech-

D. Freedman
Hewlett-Packard Laboratories, Haifa, Israel
E-mail: daniel.freedman@hp.com

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

Fig. 1 The semi-automatic segmentation paradigm. (a) The original image. (b) The user marks a small number of pixels for each of the objects present. Here, the user has used rectangles; the object seeds are marked with a blue rectangle, while the background seeds are marked with a green rectangle. (c) The resulting segmentation, using the method proposed in this paper. (d) Another visualization of the segmentation.

niques which divide the image into precisely two regions – an object and the background, the Random Walker technique is able to segment the image into an arbitrary number of objects. This is useful in many applications; for instance, in medical imaging, there are often several structures of interest in a given scan. (A more concrete example is the following: in the treatment of prostate cancer, we may be interested in four objects – the prostate, the bladder, the anterior rectal wall, and the "background," i.e. the remainder of the image.) A second nice property of the Random Walker algorithm is the modeling of the image as a graph structure. Such a structure is a handy way to encode the propagation of information described above, and it is here that we make our contribution.

In particular, we make two contributions. The first contribution pertains to the weight on the edges in the graph. Traditionally, such weights are based on the similarity between two neighbouring pixels, based on their grayscale intensities or their colours. We substitute a new measure of weights based on *probability distributions* over colours. This measure is much more robust than simple colour-based measures, as it allows for textured objects, and object that are themselves composed of more than a single perceptual component. These new weights are very effective in practice, and we will show multiple examples in which the use of the new weights leads to a substantial improvement in segmentation performance.

The second contribution concerns the graph structure itself. The traditional graph has a vertex set which is the set of pixels, and edges between each pair of neighbouring pixels. Instead of using pixels, we first oversegment the image using the Mean Shift algorithm [5–7]. We then use the resulting irregular graph, with vertices corresponding to the segments and edges corresponding to pairs of adjacent segments. This new graph is considerably smaller than the original image graph; on an image of one million pixels, it is not uncommon to find a thousand such segments. As the Random Walker algorithm much solve a (sparse) linear system which is of order the number of vertices, this can lead to a major savings in computing time. Furthermore, the Mean Shift algorithm is quite effective at oversegmentation, leading to an extra degree of robustness with respect to image noise. Indeed, one might argue that the use of this Mean Shift based graph enhances the segmentation to the extent that, in terms of the experimental results, the two contributions are fundamentally intertwined.

1.1 Paper Outline

The remainder of this paper is organized as follows. In Section 2, we review the state of related work within the field of semi-automatic segmentation. We then turn, in Section 3, to the algorithm itself. We spend the majority of the space focused on the new definition of edge weights, and issues relating to the computation of these weights in practice. In Section 4, we show results which qualitatively demonstrate the improvements in speed and accuracy that arise from the two contributions. We also run a quantitative comparison with five state of the art algorithms on the Grabcut Database [2], and show that our method outperforms these algorithms. We conclude in Section 5.

2 Related Work

Semi-automatic segmentation, which is also referred to as interactive or seeded segmentation, has been a subject of interest to the computer vision and image processing communities for the last two decades. The Intelligent Scissors and Livewire work of Mortensen and Barrett [8–10] were amongst the earliest methods to tackle the problem. They used a contour-based method, in which a curve drawn by the user would automatically "snap" to nearby image edges. The Image Snapping method of Gleicher [11] is in a similar vein.

The idea of performing interactive segmentation using Graph Cuts was introduced by Boykov and Jolly [12, 13], though it was anticipated in earlier work by Greig *et al.* [14]. The user marks a small number of background and foreground pixels as "seeds;" the goal is then to propagate the information contained in these seeds to the rest of the image. A binary energy function is formulated, in which each

pixel can have a label of either 0 (background) or 1 (foreground); the energy tries to respect images edges. That is, adjacent pixels on opposite sides of a high contrast edge are not penalized for having opposite labels; whereas adjacent pixels in a smooth region are. A global minimum of the energy can be found by combinatorial means, in particular, by solving a max-flow/min-cut problem.

There have been a number of extensions of the Graph Cuts methodology for semi-automatic segmentation. Li *et al.* [15] presented a method combining watershed segmentation and Graph Cut minimization to build an interactive image cut-out system. Lombaert *et al.* [16] improved the performance of the Graph Cut technique with a multilevel banded heuristic for computation of Graph Cuts that is motivated by the well-known narrow band algorithm in level set computation. The "GrabCut" technique of Rother *et al.* [2] introduced better colour modeling, as well as an extra layer of (local) minimization to the Graph Cuts technique, with good results. Freedman and Zhang [17] incorporated shape priors within the Graph Cuts framework. A number of papers [18–20] attempted to merge the Active Contour and Graph Cuts paradigms.

The Random Walker technique of Grady and collaborators, which we make use of in this paper, is the other major technique for propagating information from seed pixels to the rest of the image. This work, which initially appeared in [3,1], computes the probability that a random walk process from a given pixel ends up at any of the seed pixels. The transition probabilities between pixels vary inversely with the contrast between pixels, so that the walks tend not to cross edges. An advantage of this technique over the graph cuts formalism is that it can naturally handle more than two objects, which is not the case with Graph Cuts.

Extensions to the Random Walker technique include the incorporation of priors [4], a faster technique which precomputes eigenvectors [21], and an extension to the case of random walks on directed graphs [22]. In addition, interesting connections [23,24] have been noted between the Graph Cuts technique, the Random Walker method, algorithms based on shortest paths [25], and watersheds [26]. There is also related work in the area of learning-style transduction [27].

3 The Algorithm

In this section, we describe the algorithmic contributions of the paper. We begin, in Section 3.1, with a more in-depth overview of the Random Walker technique of Grady *et al.* [3, 1]. Given this framework, we then describe in Section 3.2 the principal contribution of the paper: the probability distribution based weights. We pay particular attention to implementation related issues, as well as issues related to the setting of parameters. In Section 3.3, we describe how to use Mean Shift as a preprocessing step to enable an oversegmentation, leading to a considerably smaller irregular graph which can be used for the Random Walker.

3.1 The Framework

For the semi-automatic segmentation algorithm, we will assume a particular type of user input, often referred to as scribbles or seeds. As shown in Figure 1, the user marks a few pixels in each object of interest; in Figure 1, the marks are in the form of small rectangles, though any simple form of marking may be used. It is assumed that the user has not made any mistakes – that is, the seeds are indeed pixels drawn from the various objects. Given the seeds, the goal of the semi-automatic segmentation scheme is to take the limited information provided by the user, and to propagate it to the rest of the image, so that all pixels can then be labeled as belonging to a specific object.

We will focus on the Random Walker technique of Grady *et al.* [3,1]. An undirected graph $G=(V,E)$ is formed, where the vertex set V equals the set of pixels in the image. Note that the vertex set can be partitioned into a set of "marked vertices", V_m, which are those pixels that the user has marked as belonging to the various objects, i.e. the seeds; and V_u, the "unmarked vertices," comprising the remaining set of vertices. The edge set E consists of pairs of pixels which are neighbours in the image, using either a standard or 4- or 8-neigbourhood. The graph is weighted; the weight of an edge $e=(v_i,v_j)$, which we denote either $w(e)$ or $w(v_i,v_j)$, is taken to depend on the difference between the grayscale intensities or colours of the two pixels. That is, let $d(v_i,v_j)=\|z(v_i)-z(v_j)\|$, where z is either grayscale intensity (a scalar) or colour (a 3-dimensional vector); then the weight is given by an expression like

$$w(v_i,v_j)=e^{-d(v_i,v_j)^2/\sigma^2} \text{ or } w(v_i,v_j)=\frac{1}{1+d(v_i,v_j)/\sigma}$$

where σ is a parameter to be set. Thus, the weights lie in the range $[0,1]$; for similar pixels, we will have a weight close to 1, whereas for very different pixels we will have a weight close to 0.

Given this graph structure, the idea behind the Random Walker algorithm is as follows. Suppose that there are K possible objects (including the background) within the image, and that each one of the marked vertices in V_m belongs to one of these K objects. Now, focus on a particular edge e whose endpoints are the vertices v_i and v_j, i.e. $e=(v_i,v_j)$. We imagine a random walk over the graph G: since the edge weight $w(e)$ lies in the range $[0,1]$, the weight may be interpreted as a transition probability from vertex v_i to vertex v_j. Given the above definition of the edge weights, we can see that a random walk is likely to transition from v_i to v_j if v_i is

very similar to v_j (in colour or intensity), and is unlikely to make the transition if the two vertices are dissimilar.

Now, let us consider a specific random walk on the graph, given these transition probabilities. In particular, for each unmarked vertex $v_i \in V_u$, we compute the probability that a random walk beginning at that vertex reaches *any* one of the marked vertices from a particular object k; we denote this quantity by p_i^k. We then segment the image according to these probabilities. More specifically, for any vertex v_i, we classify it as belonging to segment k if $p_i^k > p_i^{k'}$ for all $k' \neq k$. Note that edges in the image (as opposed to edges in the graph) correspond to low transition probabilities, as they involve a rapid change in colour or intensity. Thus, this algorithm will tend to respect image edges in performing the segmentation.

It turns out that the probabilities p_i^k may be computed through the solution of a large, sparse linear system. In particular, let the degree of a vertex v_i be $\delta_i = \sum_{j \in adj(i)} w(v_i, v_j)$, and let the Laplacian matrix be defined in the usual way:

$$L_{ij} = \begin{cases} \delta_i & \text{if } i = j \\ -w(v_i, v_j) & \text{if } i \text{ and } j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}$$

Further, let the vertices be sorted so that the marked vertices are on top, followed by the unmarked vertices; the corresponding block-decomposition of the Laplacian may be written

$$L = \begin{bmatrix} L_m & B \\ B^T & L_u \end{bmatrix} \tag{1}$$

That is, L_m is the $|V_m| \times |V_m|$ submatrix of L consisting of the first $|V_m|$ rows and first $|V_m|$ columns; B is the $|V_m| \times |V_u|$ submatrix of L consisting of the first $|V_m|$ rows and the final $|V_u|$ columns; and L_u is the $|V_u| \times |V_u|$ submatrix of L consisting of the final $|V_u|$ rows and final $|V_u|$ columns.

Finally, let f^k be a $|V_m| \times 1$ vector, such that $f_i^k = 1$ if vertex v_i has been marked as belonging to k^{th} object, and 0 otherwise. Then if p^k is the $|V_u| \times 1$ vector containing the variables p_i^k, it can be computed as the solution to the linear system

$$L_u p^k = -B f^k \tag{2}$$

We are therefore required to solve K such linear systems. (In fact, $K-1$ will do; see [1].) The interested reader is referred to [1] for further details.

3.2 Probability Distribution-Based Weights

For the Random Walker scheme to be successful, it is important that the graph structure itself conveys the proper notions of homogeneity of a region. As noted above, traditional weights depend on the difference in grayscale intensity or colour of an object; that is, they vary inversely with the distance

$$d(v_i, v_j) = \|z(v_i) - z(v_j)\|$$

where z is either intensity or colour. If one possesses no prior information about the objects of interest, this is perhaps the best one can do. However, we *do* possess prior information: the user input in the form of seeds gives us some important information about the profiles of the various objects. It is this extra information that we wish to exploit in formulating more meaningful edge weights.

(a) (b) (c)

Fig. 2 The need for probability distribution-based weights; see discussion in the text. (a) Synthetic texture. (b) Coral snake. (c) Elephant with heterogeneous background.

To illustrate the problem, examine the three images in Figure 2. The left image shows a combination of two synthetic textures; the middle image, a coral snake which is itself composed of three separate colours; and the right image an elephant, with a heterogeneous background. In each case, traditional edges weights, based on grayscale intensity or colour differences, will lead to small edge weights, and thus low transition probabilities, *within* perceptual segments. In the case of the synthetic textures, we can expect the small weights to appear along the boundary between the light brown veins and the dark brown blobs in the top half of the image. In the case of the coral snake, such weights would appear between the red, black, and yellow segments. In the case of the elephant, small weights would manifest themselves along the divisions between sky, mountains, bushes, and grass. In all three cases, the result will likely be a segmentation not according to the objects (or backgrounds) that we have in mind, but rather accordingly to grayscale of colour homogeneity. It is this problem that we seek to fix.

3.2.1 The Weights

As has already been noted, we have prior information on the objects of interest in the form of the user input. Focus on the image of the coral snake, and suppose that the user input is the seed (in this case, a rectangle) as shown in Figure 1.

This input gives us all the information we need to model the object, in the form of a *probability density*. In particular, denote the 3-vector for colour by z; each object k that we wish to segment has seeds, and from these seeds we can learn a probability density for that object, which we denote

$$\rho^k(z), \quad k = 1, \ldots K$$

In Section 3.2.2, we will describe the process by which the density $\rho^k(\cdot)$ is learned from the seeds for object k. For the moment, however, let us take the densities as given, and let us define edge weights using these densities.[1]

If two adjacent vertices belong to the same object, then we do not necessarily expect that their colours will match – as we have already seen from the images in Figure 2. By contrast, however, we would expect that the *probability density evaluated at their colours* should match, at least approximately. More specifically, if v is a vertex, and $z(v)$ is its colour, then $\rho(z(v))$ is the probability density evaluated at its colour. Now, if the two vertices v_i and v_j both belong to object k, we would expect that the scalars $\rho^k(z(v_i))$ and $\rho^k(z(v_j))$ should both be relatively large, as the colours $z(v_i)$ and $z(v_j)$ are both parts of the object k, and hence should be represented in that object's probability density $\rho^k(\cdot)$. Thus, even though $z(v_i)$ and $z(v_j)$ may be quite different, $\rho^k(z(v_i))$ and $\rho^k(z(v_j))$ will be quite similar. Similarly, if neither vertex belongs to object k, we would expect that $\rho^k(z(v_i))$ and $\rho^k(z(v_j))$ should both be relatively small; again, $\rho^k(z(v_i))$ and $\rho^k(z(v_j))$ should roughly match. On the other hand, if vertex v_i belongs to object k and vertex v_j does not, then $\rho^k(z(v_i))$ will be large and $\rho^k(z(v_j))$ will be small, so that $\rho^k(z(v_i))$ and $\rho^k(z(v_j))$ will not match, as desired. Thus, a first version of the distance between vertices might be

$$d^k(v_i, v_j) = |\rho^k(z(v_i)) - \rho^k(z(v_j))|$$

with the weights correspondingly given by

$$w^k(v_i, v_j) = \frac{1}{1 + d^k(v_i, v_j)/\sigma}$$

Note that according to the above expressions, the weight for a given edge is different for each object k. Thus, we must form k separate Laplacians, L^k, satisfying

$$L^k_{ij} = \begin{cases} \delta^k_i & \text{if } i = j \\ -w^k(v_i, v_j) & \text{if } i \text{ and } j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where as before, the degree of a vertex is

$$\delta^k_i = \sum_{j \in adj(i)} w^k(v_i, v_j).$$

As in the case of the ordinary Random Walker (i.e. with colour- or intensity-based weights), we must solve the Random Walker equation k separate times, one linear system for each object:

$$L^k_u p^k = -B^k f^k \tag{4}$$

where L^k_u and B^k are drawn from the block decomposition of L^k according to Equation (1). What distinguishes (4) from the ordinary Random Walker in (2) is that in (4) the Laplacian L_k, and hence its submatrices L^k_u and B^k, is different for each of the k linear systems that must be solved; in (2), a single Laplacian L is used for all k linear systems, and what varies is simply the vector f^k. Finally, as in the ordinary Random Walker, to achieve segmentation we classify a vertex v_i as belonging to segment k if $p^k_i > p^{k'}_i$ for all $k' \neq k$.

Note that this technique effectively eliminates our problem. The homogeneity of a region is now defined in terms of a probability density, rather than colours or grayscale intensities. As a result, an object which is either textured or has multiple components (as in Figure 2), will be much more homogeneous from the point of view of the learned probability density. For example, we would expect that for the case of the coral snake, the value of the probability density will not change very much as we move from the red to the yellow regions, at least as compared to the change in the colours themselves. As a result, we will not have the same problem of very low edge weights along the boundary between red and yellow regions.

In fact, we can do even better. Rather than use the densities, consider the following quantity:

$$\pi^k_i = prob(v_i \in obj_k | z(v_i) = z),$$

that is, the *conditional probability* that vertex i is part of the k^{th} object given that we know its colour is z. Based on this quantity, we may compute the distance and the weights as

$$d^k(v_i, v_j) = |\pi^k_i - \pi^k_j| \tag{5}$$

and

$$w^k(v_i, v_j) = \frac{1}{1 + d^k(v_i, v_j)/\sigma} \tag{6}$$

Why move from densities to conditional probabilities? There are two advantages to working with the quantity π^k_i rather than the density-based quantity $\rho^k(z(v_i))$:

1. Probabilities are bounded between 0 and 1, which is not the case with densities. As a result, it is easier to set the parameter σ in Equation (6). In fact, we simply take a value of $\sigma = 0.1$ in all of our experiments. (Since densities can take on very large values, the setting of σ could become a bit tricky if densities were used.)

[1] The assumption that the learned density matches the true density of the object will depend, of course, on the choice the user has made for the seeds. In fact, it is not critical for the learned density to exactly match the true density, but rather for the *supports* of the two densities to roughly correspond. This will become clearer in our discussion of the conditional probabilities π^k_i, later in this section.

2. Within a given object, probabilities tend to be even smoother than densities. To see why, imagine an example in which the first object is a mixture of green and red, in proportions of 70 % and 30 %, respectively; whereas the second object is blue. Then within the first object, the density will be larger for green than for red, leading to variations in the value of $\rho^k(z(v_i))$ as we transition from green to red pixels, and hence to edge weights (on edges spanning green and red pixels) which are not close to 1. By contrast, it is easy to see that there will be *no* variation in the value of π_i^k as we transition from green to red pixels, and hence that the corresponding edge weights will all be 1, as desired.

There still remains the issue of computing π_i^k given our knowledge of the densities $\rho^k(z(v_i))$. This is easily accomplished using Bayes' Rule:

$$\begin{aligned}\pi_i^k &= prob(v_i \in obj_k | z(v_i) = z) \\ &= \frac{prob(z(v_i) = z | v_i \in obj_k) prob(v_i \in obj_k)}{\sum_{k'=1}^K prob(z(v_i) = z | v_i \in obj_{k'}) prob(v_i \in obj_{k'})} \\ &= \frac{\rho^k(z(v_i))}{\sum_{k'=1}^K \rho^{k'}(z(v_i))} \end{aligned} \tag{7}$$

where in the last line, we have assumed $prob(v_i \in obj_{k'}) = 1/K$ for all k'. This is a reasonable assumption in the absence of other information.

The overall segmentation scheme is then described in Figure 3.

Segmentation Scheme

Learn Densities: Learn probability densities $\rho^k(z)$ for each object k from the seeds for that object, using the procedure described in Section 3.2.2.
Conditional Probabilities: Compute conditional probabilities π_i^k for each vertex v_i and each object k, according to Equation (7).
Edge Weights: Compute edge weights according to Equations (5) and (6).
Laplacians: Compute the k Laplacian matrices and their block decompositions according to Equations (3) and (1).
Random Walker Probabilities: Compute the random walker probabilities by solving the sparse linear system (4).
Segmentation: For any vertex v_i, classify it as belonging to segment k if $p_i^k > p_i^{k'}$ for all $k' \neq k$.

Fig. 3 Segmentation scheme with the new edge weights.

Finally, note that it is possible to use the conditional probabilities π_i^k as prior probabilities. That is, if π_i^k is large, we might like to bias the segmentation towards choosing vertex i as part of segment k. If desired, such priors can be incorporated in a straightforward fashion into the Random Walker framework, see [4].

3.2.2 Learning the Densities from Seeds

We may now turn to the issue of learning the densities $\rho^k(z)$ from the user inputs, which in turn can be used to compute the conditional probabilities π_i^k. In effect, for each object k we are given a set of samples $\{z_i^k\}_{i=1}^{n_k}$ which are the colour vectors of the seed pixels for object k. Based on this set of colour vectors, we wish to compute a density estimate $\rho^k(z)$.

The problem of nonparametric density estimation has been extensively treated in the literature, see for example [28]. In general, there are two common approaches: histograms, and kernel density estimates (KDEs). Histograms have the advantage of being a compact representation; that is, they have low space complexity. Their main disadvantage is that they are not smooth. KDEs are the opposite case: they are smooth, and as a result, have better asymptotic convergence properties than histograms, but their space complexity is not good. In particular, KDEs require $O(n)$ space, where n is the number of samples used to generate the estimate. A histogram's space complexity is, instead, the number of non-zero bins, which does not depend directly on n.

Instead of using either histograms or KDEs, we use a combined representation which has the best of both worlds: a small space complexity as well as smoothness. Such representations have certainly been proposed before, as for example in [29]; nonetheless, we present our own version here, as it is particularly simple to implement. Note that we work in the CIE 1976 (L*,u*,v*) colour space [30], which we refer to informally as Luv. We could in principle work in any colour space, but as is well known, Luv is a more perceptually meaningful colour space than is RGB.

The scheme is straightforward: the essence involves computing a histogram of the data, and then interpolating between bins to gain smoothness. More specifically, suppose that the bins are denoted by index b; the probability of each bin, as given by the histogram, is ξ_b; the midpoint of each bin is m_b; and an arbitrary colour vector z falls in bin $b(z)$. Furthermore, since the bin structure is a hyperrectangle, then each bin may be written as $b = (i_1, i_2, i_3)$, i.e. one coordinate for each dimension of the colour vector.

Now, suppose that we are dealing with a vector z which falls into bin $b(z)$; to compute the density $\rho(z)$, we will interpolate the neighbouring bin probabilities $\xi_{b'}$, where the neighbours b' are those which satisfy $\|b' - b(z)\|_\infty \leq 1$. Since the colour vector is three-dimensional, this is a neighbourhood with 27 elements, including the bin $b(z)$ itself. Write $b(z) = (i_1^z, i_2^z, i_3^z)$; we then write the interpolation as

$$\rho(z) = \frac{1}{C} \sum_{i_1=-1}^{1} \sum_{i_2=-1}^{1} \sum_{i_3=-1}^{1} \omega_{i_1,i_2,i_3}(r(z)) \xi_{i_1^z+i_1, i_2^z+i_2, i_3^z+i_3}$$

In the above expression, $r(z)$ is a relative coordinate within bin $b(z)$, given by

$$r(z) = \frac{z - m_{b(z)}}{s}$$

where s is the width of the bin; and $\omega_{i_1,i_2,i_3}(r(z))$ is the weight given to the neighbour whose index is $(i_1^z + i_1, i_2^z + i_2, i_3^z + i_3)$.

We require that the weights ω sum to 1

$$\sum_{i_1=-1}^{1} \sum_{i_2=-1}^{1} \sum_{i_3=-1}^{1} \omega_{i_1,i_2,i_3}(r) = 1 \quad \forall r$$

and C is a normalizing constant to be defined shortly. To satisfy the constraint of summing to 1, while also ensuring that the density estimate is sufficiently smooth, i.e. $\rho(\cdot) \in C^2$, we define

$$\omega_{i_1,i_2,i_3}(r) = \prod_{j=1}^{3} \omega_{i_j}(r_j)$$

where the components of the vector r are r_j, i.e. $r = (r_1, r_2, r_3)$, and

$$\omega_{-1}(\gamma) = \max\left\{\frac{1}{2}[1-(1+2\gamma)^3], 0\right\}$$
$$\omega_{+1}(\gamma) = \max\left\{\frac{1}{2}[1-(1-2\gamma)^3], 0\right\}$$
$$\omega_0(\gamma) = 1 - (\omega_{-1}(\gamma) + \omega_{+1}(\gamma))$$

where γ is a scalar argument. We illustrate the interpolation weights for a two-dimensional setting, i.e. $\omega_{i_1,i_2}(r)$ for $r = (r_1, r_2)$, in Figure 4.

The constant C should be chosen so that ρ integrates to 1, so that it is a true probability density. However, since ρ is only used to compute the conditional probabilities π_i^k, computing C is unnecessary since it will just drop out of Equation (7) in any case.

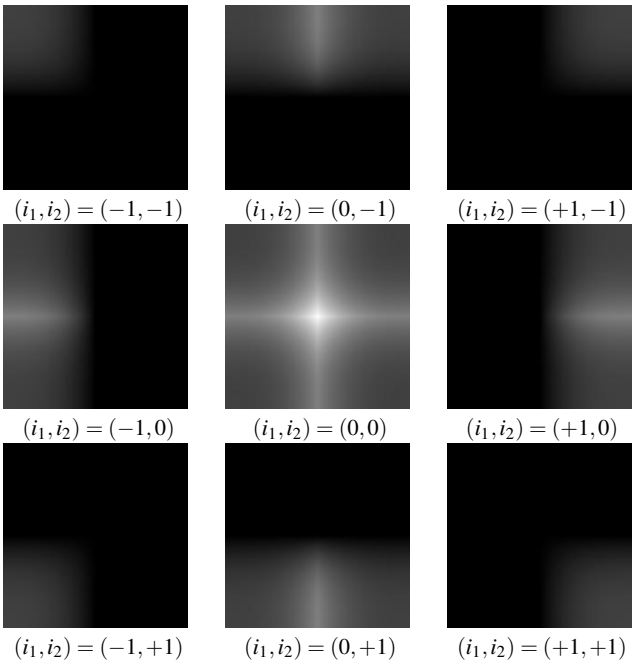


Fig. 4 The interpolation weights illustrated in a two-dimensional setting, i.e. $\omega_{i_1,i_2}(r)$. In each case, the domain of the weights is $r \in [-0.5, 0.5]^2$.

3.3 Improved Speed Using Mean Shift

Note that in order to use the Random Walker algorithm, we are required to solve the linear system given in Equation (4) K times, where K is the number of objects to be segmented. This linear system is $O(n)$ in size, where n is the number of pixels. Standard methods for solving linear systems have complexity $O(n^3)$. Due to the sparse nature of the system, this can be accelerated considerably, but the final complexity will still depend on n. In order to speed up the algorithm, therefore, it would be useful if we could somehow shrink the size of the relevant linear system. We can achieve such a shrinkage by using the Mean Shift algorithm [6,7] as a preprocessing step, to gain an initial oversegmentation.

We briefly review the Mean Shift algorithm. The pixels are taken to be samples of a probability distribution over colour space, and an estimate of the probability density is formed by using a Kernel Density Estimate (KDE). This non-parametric estimate essentially places a small bump (the kernel) at each point, and then sums up these bumps; the kernel can be taken to be a Gaussian, although kernels with compact support also exist. Given a KDE, the Mean Shift algorithm is essentially a hill-climbing algorithm; that is, starting at any point z in colour space, the Mean Shift algorithm is an efficient iterative scheme which brings z up the hill of the KDE, finally stopping at the local maximum, or mode, of the KDE in whose basin of attraction z lies. Full details of the algorithm may be found in [5–7].

It is now a simple matter to construct a smaller graph using the Mean Shift algorithm for preprocessing. The new graph, $G' = (V', E')$, has as vertices each of the segments computed with by Mean Shift. The edge set E' consists of pairs of segments which are adjacent in the plane. This graph is often called a *Region Adjacency Graph*, see Figure 5 for an illustration. This graph is irregular; fortunately, however, it still possesses the key property we require, namely that it is sparse. The fact that it is sparse follows directly from the properties of planar graphs: a planar graph with n vertices can have at most $3n-6$ edges [31]. This fact itself follows directly from the fact that the Euler Characteristic of a planar graph is 2, and that the Euler Characteristic for a two-dimensional simplicial complex is just $\chi = |V| - |E| + |F|$,

where $|F|$ is the number of faces, or triangles of the complex [32]. A little bit of algebra gives the result.

To return to the main thread of the argument, note that the irregularity of the Region Adjacency Graph does not affect the Random Walker algorithm, which does not assume a regular structure on the graph. In practice, by using this technique, the vertex set may be reduced from a set of 10^6 elements to one of 10^3 elements, leading to a huge savings in computation time. In particular, the main step in the Random Walker algorithm – the solution of a sparse linear system – has complexity which is practically linear in the number of vertices; we would therefore expect a speed-up factor of roughly the factor decrease in the size of the vertex set. Of course, this speed-up calculation is not entirely correct: it does not take into account the time to compute the Mean Shift segmentation. Nevertheless, the Mean Shift preprocessing should not be too onerous from the speed point of view, as there are a number of fast versions of Mean Shift [33–35], which have low complexities, both in theory and in practice.

We note that one may also run other oversegmentation algorithms in place of Mean Shift, such as watersheds [26] or the graph-based algorithm of Felzenszwalb and Huttenlocher [36]. Indeed, Li *et al.* [15] and Marcotegui *et al.* [37] both use watershed segmentation as the finest level of detail of their semi-automatic segmentation schemes.

It is also important to note the downside of using the Region Adjacency Graph based on Mean Shift. In particular, the Mean Shift segmentation is itself based on constructing a KDE in colour space. As a result, depending on the value chosen for the colour bandwidth (the parameter in Mean Shift which controls the width of each bump in the KDE), weak boundaries of the image may potentially be destroyed by Mean Shift. This effect is certainly noticeable in some experimental results; nonetheless, we have found that by and large it does not lead to significant issues. We now present both qualitative and quantitative experimental evaluations of the algorithm, which show that the despite this issue, the overall algorithm – consisting of both the probability distribution-based weights and the reduced graph – performs well.

4 Results

4.1 Qualitative Experimental Evaluation

In this section, we present a qualitative experimental evaluation of our algorithm. We have run the algorithm on 10 example images; in each case, we compare the results of the proposed algorithm with those that result from using colour-based weights. All images are shown in Figure 6. Column (a) shows the original image; column (b) shows the user input; columns (c) and (d) show the results of using colour-based weights, both in terms of the original image and a pure segmentation map (the latter aids with visualization); and columns (e) and (f) show the results using the probability-based weights, in a similar format.

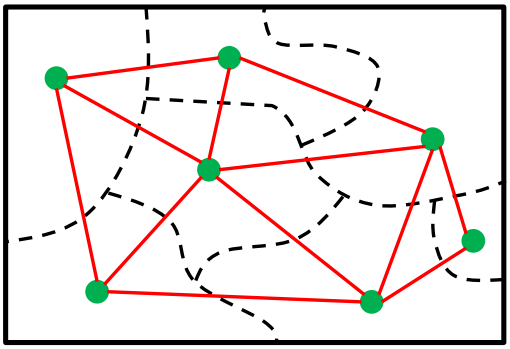


Fig. 5 The Region Adjacency Graph derived from the Mean Shift oversegmentation. The boundaries of the Mean Shift segments are denoted with dashed curves. The resulting graph is shown with vertices in green and edges in red.

Before describing the results, a few points should be noted about the experiment. The user input was in the form of rectangles, where the user selected one rectangle per object (foreground and background). Of course, other scribble-type inputs are possible, but rectangles were used for their simplicity.[2] In all cases, the Luv colour space was used. The σ parameter for the colour-based algorithm was set to 100; for the probability-based algorithm, it was set to 0.1. Both values have been chosen to produce the best results attainable, based on a visual inspection of the results on the 10 images. The number of bins for the probability-based method is taken to be 5 in each dimension (L, u, and v).

Let us now turn to analysis of the results, as shown in Figure 6. The top two rows are synthetic texture images, designed simply as a first test of the method. Note that the probability-based technique produces almost perfect results, splitting the two textures in half; the colour-based method, by contrast, makes mistakes, particularly in the second example, where a good chunk of the righthand texture is included in the lefthand segment. This is very logical, as the textures themselves possess strong colour gradients, so that a purely colour-based technique is bound to fail.

The third and fourth rows show examples in which the foreground – the coral snake and the parrot – are themselves made up of multiple colours and textures. In these cases, the probability-based method does very well, while the colour-based method fails almost completely. Note that

[2] In each of the example images shown, one seed rectangle was used per object; however, in general there is nothing to prevent the user from choosing multiple seed rectangles for a given object. This may be particularly useful if the object is composed of multiple components, or has holes. There is no change to the underlying algorithm in this case, and indeed the implementation we use will, without modification, accept multiple rectangles per object.

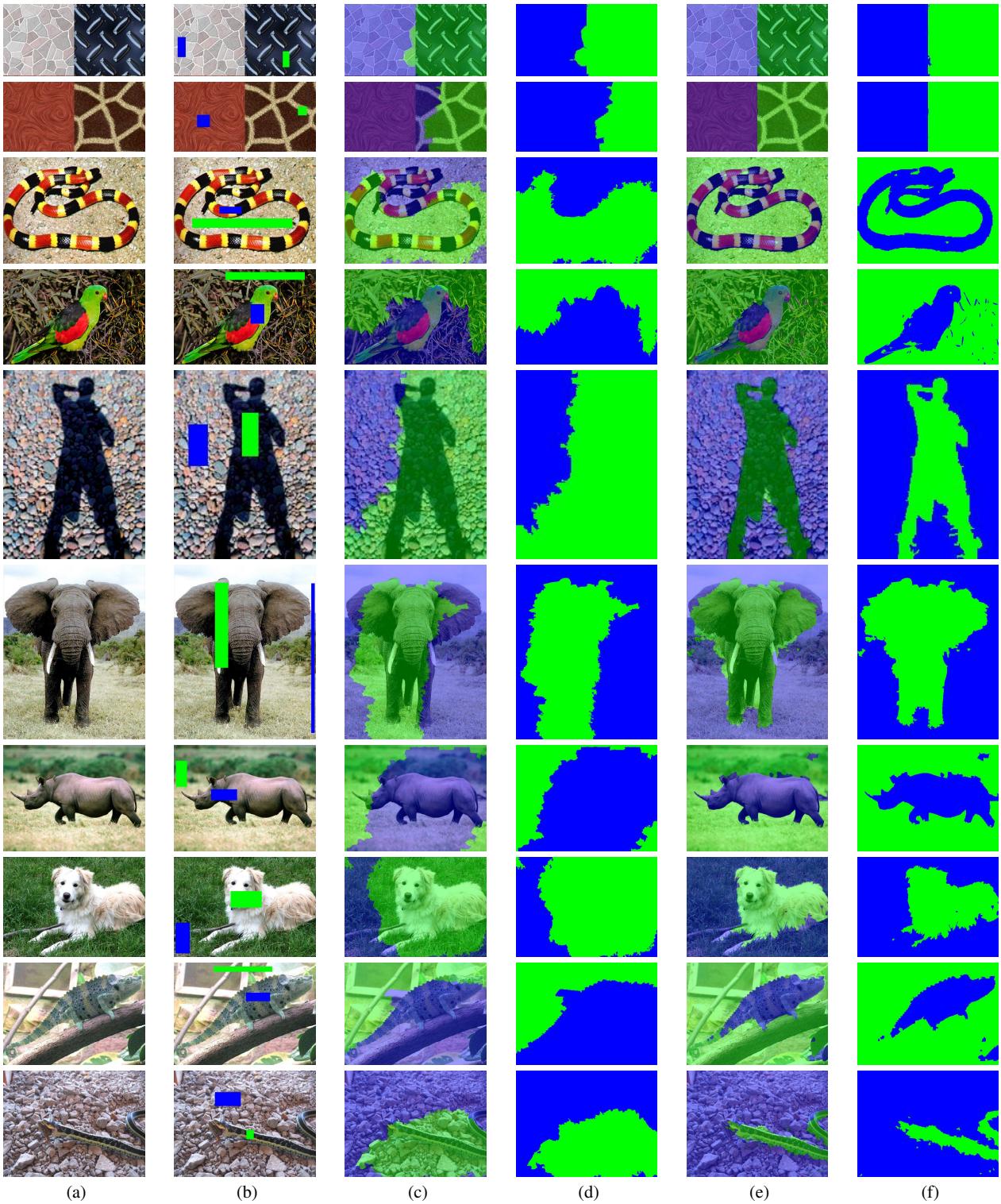
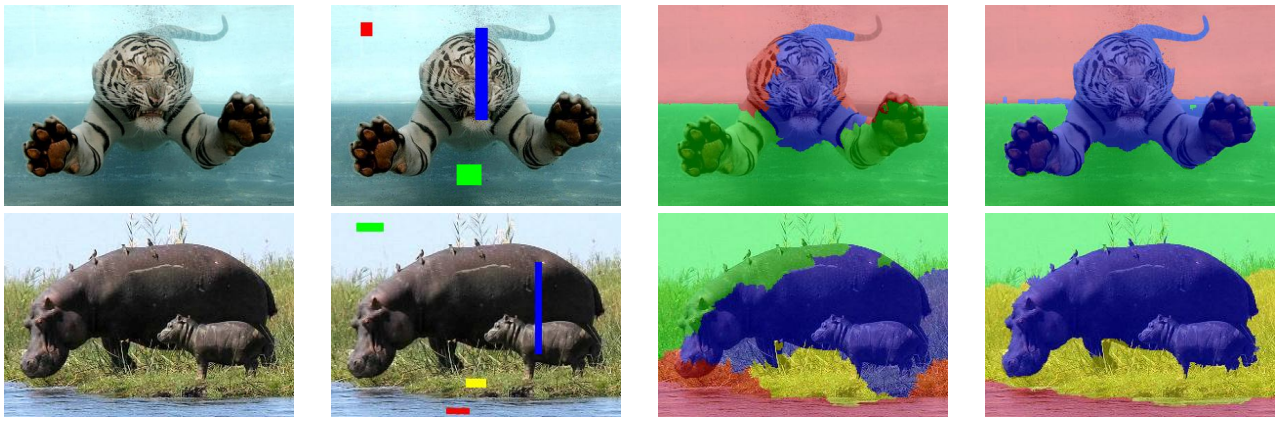


Fig. 6 (a) The original image. (b) The user input. (c) The segmentation with colour-based weights. (d) The segmentation map with colour-based weights. (e) The segmentation with probability-based weights. (f) The segmentation map with probability-based weights.

in both cases, the probability-based method does make small mistakes, although the mistakes in these cases could probably be corrected with simple post-processing, such as a morphological filter.

The examples in rows 5, 6, and 7 demonstrate situations in which the background is composed of multiple colours and textures. The background in row 5 has, in some sense, a single texture (though multiple colours); whereas the back-



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

Fig. 7 (a) The original image. (b) The user input. (c) The segmentation with colour-based weights. (d) The segmentation with probability-based weights.

grounds in rows 6 and 7 show multiple textures and colours. The probability-based segmentations in these cases show a few more artifact than those in the previous rows. In particular, the segmentation of the elephant is not perfect, which can be explained by the more complex background and foreground. The examples in rows 8, 9, and 10 continue with textured foregrounds and backgrounds, and the results are quite similar to those of the prior three rows. In all cases, the segmentations are quite good compared to the results of the colour-based scheme, which are not very informative. In particular, the colour-based scheme predictably has trouble with the textured regions, as these regions contain a lot of variation in colour, even between neighbouring vertices. As a result, the edges between neighbouring vertices within a given segment – for example, the rocky background in row 5 or the grass in row 8 – may have very small weights, contrary to what is necessary for a good segmentation.

Finally, it is worth noting the speed-up that we get from using the Mean Shift preprocessing to derive a much smaller, irregular graph, see Table 1. For each image, we note the size of the image in pixels, which is the number of vertices in the ordinary Random Walker graph (second column); and the number of segments in the Region Adjacency Graph that our algorithm uses (third column). As the solution of a sparse linear system practically has complexity which is linear in the number of vertices, the expected theoretical speed-up is the ratio of the sizes of the two graphs, which is reported in the fourth column. As has already been noted in Section 3.3, this theoretical speed-up is not the actual speed-up, due to the fact that we have ignored the time for the Mean Shift preprocessing step.

For a flavour of actual running times, including the Mean Shift preprocessing step, see Table 2. We have compared our algorithm with the implementation of the standard Random Walker which may be found on Grady's website [38]. Both methods were run on the same machine, an Intel Core 2 Duo

Image	# Pixels	# Segments	Ratio
Texture 1	45,300	269	168.4
Texture 2	44,400	207	214.5
Coral Snake	150,528	2,779	54.2
Parrot	202,400	1,378	146.9
Shadows	28,470	606	47.0
Elephant	292,200	2,110	138.5
Rhinoceros	120,000	742	161.7
Dog	252,600	2,961	85.3
Chameleon	115,600	794	145.6
Garter Snake	172,800	1962	88.1

Table 1 Decrease in the graph size through the use of Mean Shift preprocessing.

CPU, with clock speed of 2.53 GHz and 4 GB of RAM, running Windows Vista. We compared the running times for two images: one which is 512×512, the other 1024×1024. We have broken down the time in terms of the time for the Mean Shift (implemented using the fast Mean Shift technique of [35]) and the time for the remainder of the algorithm. As can be seen from Table 2, our algorithm is roughly 3 times faster than the standard Random Walker.

Image Size	Proposed Algorithm	Standard Random Walker
512×512	0.29 s (Mean Shift) + 0.63 s (remainder) = 0.92 s (total)	2.54 s
1024×1024	1.24 s (Mean Shift) + 2.82 s (remainder) = 4.06 s (total)	11.54 s

Table 2 Speed comparison: proposed algorithm vs. standard Random Walker.

Thus far, all of the examples presented have segmentations consisting of only two components. In Figure 7, we show two examples with more than two components: the swimming tiger is segmented into three components, and the

hippo is segmented into four components. In both of these examples, we see that the benefits conferred by the probability distribution-based weights over the colour-based weights carry over to the case of more than two components.

4.2 Quantitative Experimental Evaluation

In this section, we present a quantitative experimental evaluation of the algorithm. We have run the algorithm on the Grabcut Database [2], which is the standard ground truthed database for semi-automatic segmentation. The database consists of 50 images, each of which is accompanied by a ground truth (manual) segmentation. Further, the database contains seeds to be used with each image; these seeds have been derived by eroding the ground truth segmentation, according to the procedure described in [2]. An example image from the Grabcut Database, with corresponding ground truth segmentation, seeds, and segmentation provided by our algorithm, is shown in Figure 8.

Algorithm	BE	RI	GCE	VoI
Graph Cuts	3.276	0.970	0.028	0.196
Random Walker	3.206	0.972	0.026	0.185
P-Brush: $p = 1.25$	3.241	0.971	0.028	0.193
P-Brush: $p = 1.50$	3.214	0.972	0.027	0.189
P-Brush: $p = 1.75$	3.206	0.972	0.027	0.187
Ours	2.980	0.975	0.007	0.174

Table 3 Quantitative comparison of our algorithm versus the state of the art. Note that for BE, GCE, and VoI smaller is better; whereas for RI, larger is better. We outperform the five state of the art algorithms in all four criteria.

For each image, our algorithm yields its segmentation, against which we would like to compare the ground truth segmentation provided in the database. To do so, we compute four separate quantitative criteria, each of which allows for the comparison of two segmentations. We now describe each of these criteria briefly; the interested reader is directed to the references provided for a more in-depth discussion of the criteria. The Boundary Error (BE), described in [39], is a measure of the average distance between points on the curves which form the boundaries of the segmented regions. The Rand Index (RI) [40] measures the (normalized) number of agreements between the labels of pairs of pixels in the two segmentations; for example, there is an agreement if two pixels belong to the component in one segmentation, and belong to the same component in the other segmentation as well. The Global Consistency Error (GCE) [41] measures the non-overlap of segments from the two segmentations, and computes an error measure based on this. The Variation of Information (VoI) [42] is an information-theoretic measure, which is inversely related to the Mutual Information. Note that these four criteria are used in other works which quantitatively evaluate segmentation, namely [43,44].

We report the average values, across the 50 images, of each of the four criteria in Table 3. It is important to note that for BE, GCE, and VoI, small values are better; whereas for RI, a large value is better, and the highest possible value is 1. We compare our results with state of the art semi-automatic segmentation algorithms, as reported in [44] (see Table 1 in that paper). In particular, we compare the results of our algorithm with those of the Graph Cuts technique [13], the ordinary Random Walker technique with colour-based weights [1], and the P-Brush technique [44] for a variety of p-values.[3] The results are given in Table 3. We note that in terms of all four criteria, we improve on the performance of all of the five competing state of the art methods. This is perhaps more impressive given the fact that many of the images in the Grabcut Database do not contain significant amounts of texture, so that colour-based weights should be adequate.

5 Conclusions

In this paper, we have presented a technique for improving the effectiveness of the Random Walker semi-automatic segmentation scheme. In particular, we have shown how to change the underlying image graph in two ways. The first change involves a more sophisticated edge weight than the standard edge weights that are used. Instead of basing the weights on the grayscale or colour distance between neighboring vertices, we instead base it on the difference in the foreground and background probability densities. These new weights are much more robust, and allow for modeling of texture. Both qualitative and quantitative comparisons have demonstrated the effectiveness these new weights lend to the method.

Second, we have described a much more compact graph, which is achieved with the aid of the Mean Shift technique. Instead of using the standard image graph, which is regular, we substitute an irregular graph with vertices equal to the Mean Shift segments, and edges between adjacent segments. This leads to a graph which is a few orders of magnitude smaller than the original image graph, and to an algorithm which is concomitantly faster.

A future direction involves computing probability densities over more complex objects than colours. For example, we may be interested in examining texture, construed as the output of a filter bank.

[3] Note that the P-Brush technique with $p = 1$ is in fact Graph Cuts, and $p = 2$ is the ordinary Random Walker. Proving that this is the case is the central result of [44].

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

Fig. 8 An example from the Grabcut Database. (a) The image. (b) The ground truth segmentation. (c) The seeds – black and white pixels are seeds, gray pixels are not used. (d) The segmentation from the proposed algorithm.

References

1. L. Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.
2. C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 309–314, 2004.
3. L. Grady and G. Funka-Lea. Multi-label image segmentation for medical applications based on graph-theoretic electrical potentials. *Lecture Notes in Computer Science*, pages 230–245, 2004.
4. L. Grady. Multilabel random walker image segmentation using prior models. In *Proceedings of the IEEE Conference on Computer Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 763–770, 2005.
5. K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975.
6. Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.
7. D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 603–619, 2002.
8. E.N. Mortensen and W.A. Barrett. Intelligent scissors for image composition. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 191–198, 1995.
9. W.A. Barrett and E.N. Mortensen. Fast, accurate, and reproducible live-wire boundary extraction. *Lecture Notes in Computer Science*, 1131:183–192, 1996.
10. E.N. Mortensen and W.A. Barrett. Toboggan-based intelligent scissors with a four-parameter edge model. In *Proceedings of the IEEE Conference on Computer Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 452–458, 1999.
11. M. Gleicher. Image snapping. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 183–190, 1995.
12. Y. Boykov and M.-P. Jolly. Interactive organ segmentation using graph cuts. In *Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 276–286, 2000.
13. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 105–112, 2001.
14. D.M. Greig, B.T. Porteous, and A.H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, 51(2):271–279, 1989.
15. Y. Li, J. Sun, C.K. Tang, and H.Y. Shum. Lazy snapping. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, volume 23, pages 303–308, 2004.
16. H. Lombaert, Y. Sun, L. Grady, and C. Xu. A multilevel banded graph cuts method for fast image segmentation. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 259–265, 2005.
17. D. Freedman and T. Zhang. Interactive graph cut based segmentation with shape priors. In *Proceedings of the IEEE Conference on Computer Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 755–762, 2005.
18. O. Juan and Y. Boykov. Active graph cuts. In *Proceedings of the IEEE Conference on Computer Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 1023–1029, 2006.
19. N. Xu, N. Ahuja, and R. Bansal. Object segmentation using graph cuts based active contours. *Computer Vision and Image Understanding*, 107(3):210–224, 2007.
20. Y. Zeng, D. Samaras, W. Chen, and Q. Peng. Topology cuts: A novel min-cut/max-flow algorithm for topology preserving segmentation in N–D images. *Computer Vision and Image Understanding*, 112(1):81–90, 2008.
21. L. Grady and A.K. Sinop. Fast approximate Random Walker segmentation using eigenvector precomputation. In *Proceedings of the IEEE Conference on Computer Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.
22. D. Singaraju, L. Grady, and R. Vidal. Interactive image segmentation via minimization of quadratic energies on directed graphs. In *Proceedings of the IEEE Conference on Computer Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.
23. A.K. Sinop and L. Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.
24. C. Couprie, L. Grady, L. Najman, and H. Talbot. Power watersheds: A new image segmentation framework extending graph cuts, random walker and optimal spanning forest. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 731–738, 2009.
25. A. Protiere and G. Sapiro. Interactive image segmentation via adaptive weighted distances. *IEEE Transactions on Image Processing*, 16(4):1046–1057, 2007.
26. L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.
27. O. Duchenne, J.Y. Audibert, R. Keriven, J. Ponce, and F. Ségonne. Segmentation by transduction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.
28. B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, 1986.
29. M. Girolami and C. He. Probability density estimation from optimally condensed data samples. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1253–1264, 2003.
30. J. Schanda. *Colorimetry: Understanding the CIE System*. Wiley-Interscience, 2007.
31. T. Nishizeki and N. Chiba. *Planar Graphs: Theory and Algorithms*. Elsevier, 1988.

32. J.R. Munkres. Elements of Algebraic Topology. 1984.
33. C. Yang, R. Duraiswami, N.A. Gumerov, and L. Davis. Improved fast Gauss transform and efficient kernel density estimation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 664–671, 2003.
34. P. Wang, D. Lee, A. Gray, and J.M. Rehg. Fast mean shift with accurate and stable convergence. In *Proceedings of the Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2007.
35. D. Freedman and P. Kisilev. Fast Mean Shift by compact density representation. pages 1818–1825, 2009.
36. P.F. Felzenszwalb and D.P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.
37. B. Marcotegui, F. Zanoguera, P. Correia, R. Rosa, F. Marques, R. Mech, and M. Wollborn. Video object generation tool allowing friendly user interaction. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, volume 2, pages 391–395, 1999.
38. Random Walker code. `http://www.cns.bu.edu/~lgrady/random_walker_matlab_code.zip`.
39. J. Freixenet, X. Munoz, D. Raba, J. Marti, and X. Cufi. Yet another survey on image segmentation: Region and boundary information integration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 408–422, 2002.
40. C. Pantofaru and M. Hebert. A comparison of image segmentation algorithms. Technical Report CMU-R1-TR-05-40, The Robotics Institute, Carnegie Mellon University, 2005.
41. D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*.
42. M. Meila. Comparing clusterings: an axiomatic view. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 577–584, 2005.
43. A.Y. Yang, J. Wright, Y. Ma, and S.S. Sastry. Unsupervised segmentation of natural images via lossy data compression. *Computer Vision and Image Understanding*, 110(2):212–225, 2008.
44. D. Singaraju, L. Grady, and R. Vidal. P-brush: Continuous valued MRFs with normed pairwise distributions for image segmentation. In *Proceedings of the IEEE Conference on Computer Computer Vision and Pattern Recognition (CVPR)*, pages 1303–1310, 2009.