Interactive Foreground Extraction using graph cut

Carsten Rother, Vladimir Kolmogorov, Yuri Boykov, Andrew Blake

Microsoft Technical Report: MSR-TR-2011-46

Note, this is an extended version of chapter 7 from the book:
Markov Random Fields for Vision and Image Processing, MIT Press [6].
In this Technical Report, references to other chapters are with respect to the book.
The differences are, a new section 4.3 and extra details in section 3.2 and 3.3

The topic of interactive image segmentation has received considerable attention in the computer vision community in the last decade. Today, this topic is very mature and commercial products exist which feature advanced research solutions. This means that interactive image segmentation is today probably one of the most used computer vision technologies world-wide. In this chapter we review one class of interactive segmentation techniques, which use discrete optimization and a regional selection interface. We begin the chapter by explaining the seminal work of Boykov and Jolly [9]. After that the GrabCut technique [36] is introduced, which improves on [9]. GrabCut is the underlying algorithm for the Background Removal tool in the Microsoft Office 2010 product. In the third part of the chapter many interesting features and details are explained which are part of the product. In this process several recent research articles are reviewed. Finally, the Background Removal tool, as well as [9, 36], are evaluated in different ways on publicly available databases. This includes static and dynamic user inputs. [1]

1 Introduction

This chapter addresses the problem of extracting an object in an image with additional hints from the user. This is different to the long-standing research topic of automatically partitioning an image into the objects present in the scene, e.g. [38]. Firstly, user

[1]A historical note. The Background Removal tool was fully developed at the end of 2004 by Carsten Rother and Vladimir Kolmogorov. It was part of an external release of Microsoft Expression Acrylic Graphics Designer (technology preview) in June 2005 (called "smart select"). This included engineering solutions to many practically interesting problems (see sec. 4), which were not addressed in [36]. For some problems, our solutions are in fact very similar to recent work [31, 30]. Some of these practical problems motivated our recent articles on the following topics: initialization and optimality [44], connectivity [43], bounding-box prior [28], segmentation-based matting [35]. To fine-tune the Background Removal tool we employed the robot user (see sec. 5.2), which was also used in [17] and motivated our very recent work on learning interactive segmentation systems [34].

interaction is needed to specify the object of interest. Secondly, quite often the user wants to select only a part of an object, e.g. head of a person, or an arbitrary region of interest. The intrinsically interactive nature of this problem makes it very attractive, but also challenging. Hence, it has been a fruitful research topic for more than two decades, where some work concentrates more on theoretical aspects, e.g. model and optimization, and other work more on user aspects.

The question of what is the best interactive segmentation system today is hard to answer. Many factors have to be considered: i) What is the user group (e.g. novice or advanced users), ii) What is the user interface, iii) How to measure the user involvement (e.g. total amount of interaction time, or number of user hints).

It is worth mentioning that approaches for interactive image segmentation have influenced many related tasks. One example is the problem of joint object recognition and segmentation, as can be seen in the TextonBoost framework [39] or the ObjCut system [25]. Another example is the web-based retrieval system for classical vases [26], which automatically runs segmentation technology in the background.

Note that the focus of this chapter is on the binary segmentation problem, i.e. each pixel belongs to either foreground or background. This is a simplified view of the problem since some pixels, especially close to the object boundary, are semi-transparent, i.e. a mix of foreground and background colors. A brief discussion of this issue in the context of a practical segmentation system is given in sec. 4.3.

The chapter is organized as follows. After a brief literature review in sec. 1.1, three systems are presented, in the order of increased model-complexity: the Boykov and Jolly approach (sec. 2), the GrabCut system (sec. 3), and the commercial Background Removal tool (sec. 4). In sec. 5 the methods are compared to other state-of-the art techniques in two different experiments.

1.1 Interactive Image Segmentation - a Brief Review

In the following we categorize different approaches to interactive image segmentation by their methodology and user-interfaces. Note that our brief review is not meant to be comprehensive.

Magic Wand is probably the simplest technique. Given a user-specified “seed” point (or region), a set of pixels is computed which is connected to the seed point, where all pixels in the set deviate, from the color of the seed point, only within a given tolerance. Fig. 1(a) shows the result using Magic Wand in Adobe Photoshop 7 [1]. Because the distribution in colour space of foreground and background pixels have a considerable overlap, a satisfactory segmentation can not be achieved.

Intelligent Scissors (a.k.a. Live Wire or Magnetic Lasso) [32] allows a user to choose a “minimum cost contour” by roughly tracing the object’s boundary with the mouse. As the mouse moves, the minimum cost path from the cursor position back to the last “seed” point is shown. If the computed path deviates from the desired one, additional user-specified “seed” points are necessary. In fig. 1(b) the Magnetic Lasso of Photoshop 7 was used, where a large number of seed points (here 19) were needed,

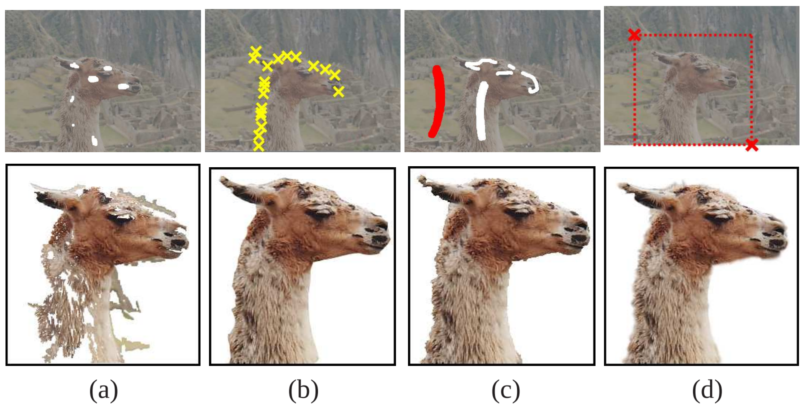


Figure 1: Foreground extraction with four different systems: (a) Magic Wand, (b) Intelligent Scissors, (c) Graph cut [9], and (d) GrabCut [36], with segmentation result in the bottom row and user interaction in the top row (image colors were changed for better visualization, see original color image in fig. 1.7(a) (chapter 1). While the results (b-d) are all visually acceptable, GrabCut needs fewest user interactions (two clicks). Note, result (d) is the final result of GrabCut including semi-transparency using border matting.

since both foreground and background are highly textured. One problem is that this technique is not effective for objects with a long boundary (e.g. a tree with many branches).

Segmentation in Discrete Domain. Boykov and Jolly were the first to formulate a simple generative MRF model in discrete domain for the task of binary image segmentation [9]. This basic model can be used for interactive segmentation. Given some user constraints in the form of foreground and background brushes, i.e. regional constraints, the optimal solution is computed very efficiently with graph cut, see an example in fig. 1(c). The main benefits of this approach are: global optimality, practical efficiency, numerical robustness, ability to fuse a wide range of visual cues and constraints, unrestricted topological properties of segments, and applicability to N-D problems. For these reasons, this approach inspired many other methods for various applications in computer vision, see e.g. chapter 9on bilayer segmentation in video. It also inspired the GrabCut system [36, 7], which is the main focus of this chapter. GrabCut solves a more challenging problem, namely the joint optimization of segmentation and estimation of global properties of the segments. The benefit is a simpler user interface in form of a bounding box, see example in fig. 1(d). Note, such joint optimization have been done in other contexts before. An example is depth estimation in stereo images [4] where the optimal partitioning of the stereo images and the global properties (affine warping) of each segment are optimized jointly.

Since the work of Boykov and Jolly, many articles on interactive segmentation using graph cut and a brush interface have been published; a few are [29, 16, 15, 44, 2, 40, 30, 34, 17] which we will discuss in more detail later. We would like to refer to chapter 8where the discrete labeling problem is relaxed to a continuous one, which gives a common framework for explaining and comparing three popular approaches, random walker [16], graph cut [9] and geodesic distance [2]. Another interesting set of discrete functionals are based on ratio, e.g. area over boundary length, see e.g. [14, 19, 23].

Segmentation in Continuous Domain. There are very close connections between the spatially, discrete MRFs, as mentioned above, and variational formulations in the continuous domain. The first continuous formulations were expressed in terms of snakes [20] and geodesic active contours [12], related to the well-known Mumford-Shah functional [33]. The goal is to find a segmentation that minimizes a boundary (surface) under some metric, typically image-based Riemannian metric. Traditionally, techniques such as level-sets were used, which however are only guaranteed to find a local optimum. Recently, many of these functionals were reformulated, using convex relaxation, i.e. the solution lives in the $[0,1]$ domain, which allows to achieve global optimality and bounds in some practical cases (see chapter 12). An example for interactive segmentation with a brush-interface is [42], where the optimal solution of a weighted TV-norm is computed efficiently. Instead of using convex relaxation techniques, the continuous problem can be approximated on a discrete grid and solved, globally optimally, using graph cut. This can be done for a large set of useful metrics, see [10, 22]. Theoretically, the discrete approach is inferior since the connectivity of the graph has to be large in order to avoid metrication artifacts. In practice, however, artifacts are rarely visible when using a *geodesic* distance, see e.g. fig. 1(d) with an underlying 8-connected graph. In sec. 3.1 we will give another relationship between the continuous Chan-Vese functional [13] and the discrete GrabCut functional.

Paint-Selection. Conceptually the brush interface and the so-called “paint-selection” interface [30] are very similar. The key difference is that a new segmentation is visualized after each mouse movement, i.e. instant feedback while drawing a stroke. In sec. 4 a more detailed comparison with the Background Removal tool is given.

2 Basic Graph Cut model for Image Segmentation

Boykov and Jolly [9] addressed the problem of interactive image segmentation based on initial trimap $T = \{T_F, T_B, T_U\}$. The trimap partitions the image into three sets: T_F and T_B comprises of pixels selected by the user as either foreground or background respectively, and T_U is the remaining set of unknown pixels. The image is an array $\mathbf{z} = (z_1, \ldots, z_n, \ldots, z_N)$ of intensities (grey, color, or any other n-dimensional values), indexed by integer n. The unknown segmentation of the image is expressed as an array of “opacity” variables $\mathbf{x} = (x_1, \ldots, x_N)$ at each pixel. In general, $0 \leq x_n \leq 1$ (e.g. α-matting), but [9] use discrete-valued (hard) segmentation variables $x_n \in \{0,1\}$, with 0 for background and 1 for foreground. The parameter ω describes the distributions for foreground and background intensities. The basic approach in [9] assumes that such distributions (intensity models or histograms $\omega = \{h_B(z_i), h_F(z_i)\}$ for foreground and background) are either known a priori, or assembled directly from labelled pixels from the respective trimap regions T_B, T_F. Histograms are normalized to sum to 1 over the range of intensities, i.e. $\int_z h_F(z) = 1$. This means that the histograms represent the observation likelihood, i.e. $P(z_i|x_i = 0) = h_B(z_i)$ and $P(z_i|x_i = 1) = h_F(z_i)$.

The segmentation task addressed in [9] is to infer the unknown opacity variables $\mathbf{x}$ from the given model ω and image data $\mathbf{z}$. For this, an energy function E is defined so that its minimum should correspond to a good segmentation, in the sense that it is guided both by the given foreground and background intensity histograms and that the opacity is "coherent", reflecting a tendency to solidity of objects. This is captured by a "Gibbs" energy of the form:

$$E(\mathbf{x}, \omega, \mathbf{z}) = U(\mathbf{x}, \omega, \mathbf{z}) + V(\mathbf{x}, \mathbf{z}) \ . \tag{1}$$

The data term U evaluates the fit of the segmentation $\mathbf{x}$ to the data $\mathbf{z}$, given the model ω, and is defined for all pixels in T_U as:

$$U(\mathbf{x}, \omega, \mathbf{z}) = \sum_{n \in T_U} -\log h_B(z_i)[x_n = 0] - \log h_F(z_i)[x_n = 1] + \sum_{n \in T_F \cup T_B} H(x_n, n) \tag{2}$$

where $[\phi]$ denotes the indicator function taking values $0, 1$ for a predicate ϕ, and the term $H(x_n, n)$ constrains certain variables to belong to foreground or background respectively, i.e. $H(x_n, n) = \gamma([x_n = 0][n \in T_F] + [x_n = 1][n \in T_B])$, where γ is a large enough constant. The smoothness term can be written as

$$V(\mathbf{x}, \mathbf{z}) = \sum_{(m,n) \in \mathcal{N}} dis(m,n)^{-1} \ (\lambda_1 + \lambda_2 \exp\{-\beta ||z_m - z_n||^2\}) \ [x_n \neq x_m], \tag{3}$$

where $\mathcal{N}$ is the set of pairs of neighboring pixels, and $dis(\cdot)$ is the Euclidean distance of neighbouring pixels This energy encourages coherence in regions of similar intensity-level. In practice, good results are obtained by defining pixels to be neighbours if they are adjacent either horizontally/vertically or diagonally (8-way connectivity). Note that factor $dis \cdot ()$ and larger neighborhoods help the smoothness term $V(\mathbf{x}, \mathbf{z})$ to better approximate a geometric length of the segmentation boundary according to some continuous metric, see [10]. This reduces geometric artifacts. When the constant λ_2 is set to 0, the smoothness term is simply the well-known Ising prior, encouraging smoothness everywhere. Practically, as shown in [9], it is, however, far more effective to set $\lambda_2 > 0$ as this relaxes the tendency to smoothness in regions of high contrast. The constant β is chosen to be: $\beta = \left(2 \left\langle (z_m - z_n)^2 \right\rangle \right)^{-1}$, where $\langle \cdot \rangle$ denotes expectation over an image sample. This choice of β ensures that the exponential term in (3) switches appropriately between high and low contrast (see [8]). The constants λ_1 and λ_2 should be learned from a large corpus of training data. Various different learning approaches have been suggested in the past, ranging from simple cross-validation [8], over max-margin learning [41], and very recently parameter estimation in an interactive setting [34]. In most of our experiments the values were fixed to the reasonable choice of $\lambda_1 = 5$ and $\lambda_2 = 50$.

Now that energy (1) is fully defined, the Boykov&Jolly [9] model for binary segmentation can be formulated as estimation of a global minimum

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \ E(\mathbf{x}, \omega, \mathbf{z}). \tag{4}$$

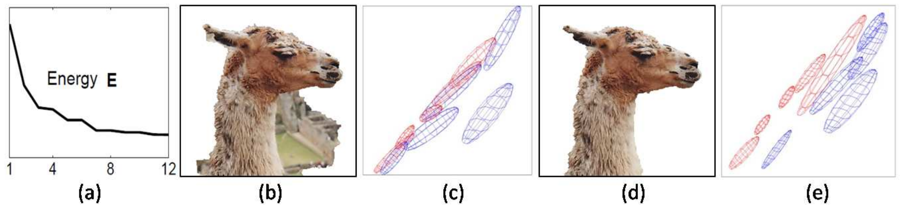


Figure 2: (a) The energy of the GrabCut+ model decreases over 12 iterations. (b) Initial result after first iteration of the GrabCut+ algorithm, where initialization is as in fig. 1(d). (c) The GMMs, here $K = 5$, for foreground (blue) and background (red) do overlap considerably (visualized RG-slice). (d,e) The final result for segmentation and GMMs.

Exact global minima can be found using a standard minimum cut/maximum flow algorithm [11] (chapter 2). Since the desired results are often not achieved with the initial trimap, additional user interactions are necessary. The maximum flow computation for these additional interactions can be made very efficient by reusing flow from the previous computation (see details in [9]).

3 GrabCut+ : Image Segmentation using iterative Graph Cut

The following description of GrabCut contains additional details, and a few minor modifications, compared to the original version [36, 7], hence the name GrabCut+ is used.

The algorithm described in the previous section often gives in practice good results, as shown in fig. 1(c), however, it fully relies on the user to define the color distributions for foreground and background. One problem is that it does not exploit the information given by the unlabeled data to learn, or infer, the unknown parameter ω in a better way. In the following we describe one approach which makes use of the unlabeled data. The simple idea is to find jointly the optimal settings for the model parameter ω and segmentation $\mathbf{x}$. This is done as before by minimizing the functional in (1) subject to the given user constraints. Note that by optimizing ω we implicitly assume that both foreground and background are represented well by compact distributions. The implications of this assumption are discussed later in detail. By exploiting the unlabeled data we are able to achieve good results with fewer user inputs compared to the previous approach in sec. 2. In particular we show that it is sometimes sufficient to simply specify the object with a bounding box, i.e. the set T_F is empty, (see example in fig. 1(d)). Unfortunately, optimizing this energy with respect to both unknowns, ω and $\mathbf{x}$, is a very challenging problem, in fact it is NP-hard [44], as discussed later. Hence, in this section and the following one, questions concerning different optimization procedures, optimality, and initialization are addressed.

3.1 The GrabCut Model

The first modification of the basic segmentation model, as described in sec. 2, is done by switching from an explicit representation of intensity distributions via histograms to a parametric representation via Gaussian Mixture Models (GMMs) [36, 8, 37]. This more compact form of representation is particularly helpful in case of RGB colors (n-dimensional intensities).

Foreground and background are modeled separately with each K full-covariance Gaussian (here $K = 7$). In order to deal with the GMM tractably, in the optimization framework, an additional vector $\mathbf{k} = \{k_1, \ldots, k_n, \ldots, k_N\}$ is introduced, with $k_n \in \{1, \ldots K\}$, assigning, to each pixel, a unique GMM component, one component either from the foreground or the background model, according to $x_n = 1$ or 0.[2] This means that the unknown parameter ω comprises the variables

$$\omega = \{\mathbf{k}, \pi_F(k), \mu_F(k), \Sigma_F(k), \pi_B(k), \mu_B(k), \Sigma_B(k),\ k = 1 \ldots K\},$$

with π as mixture weighting coefficients, which sum up to 1, and $\mu(k), \Sigma(k)$ as mean and covariance matrix for each Gaussian k.[3] It is important to note that fitting a GMM model is strictly speaking an ill-posed problem since fitting a Gaussian to the color of a single pixel gives an infinite likelihood (see [5] sec. 9.2.1). Hence, the covariance matrix is restricted to have a minimum variance, e.g. $1/255^2$.

Using the same energy (1), the GrabCut model is defined as the joint optimization (estimation) for segmentation x and parameters ω

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \ \min_{\omega} E(\mathbf{x}, \omega, \mathbf{z}). \tag{5}$$

The key difference between the Boykov&Jolly model (4) and the GrabCut model (5) is that in (5) the minimization is also done with respect to ω. It is worth noting that the GrabCut model and the functional of Chan-Vese [13, 27] in continuous domain, related to the Mumford-Shah functional [33], share some properties. In both models the key problem is the joint optimization of segmentation and global properties of the segmented regions.

3.2 The Optimization Procedure

The pseudo-code for GrabCut+ is given in fig. 3. The user starts by defining the initial trimap using either a bounding box or lasso interface. This means that T_B is outside the marked region, T_U inside the marked region and T_F is an empty set. As suggested in [36] results improve if T_B only comprises pixels which are inside a strip around the outside of the marked region [4]. The intuition is that the relevant background training data is often close to the object. In fact, pixels outside this strip are ignored throughout

[2]Using "soft assignments" of probabilities for each component to a given pixel would give a significant additional computational expense for a negligible practical benefit.

[3]Note, an efficient variant for using GMMs in a segmentation framework has been suggested in [39]. A different GMM with $2K$ Gaussian is fitted first to the whole image. This gives a fixed assignment vector $\mathbf{k}$, which is not updated during the optimization of ω and $\mathbf{x}$.

[4]The width of the strip is chosen as a small fraction of the bounding box dimensions. In the experiments, sec. 5.1, the width is set to 10 pixels as in [28].

Algorithm 1 : GrabCut+

```
Require: $T_B$ using bounding box or lasso user interface. Set $T_F = \emptyset, T_U = \overline{T_B}$
1: Initialize $x_n = 0$ for $n \in T_B$ and $x_n = 1$ for $n \in T_U$
2: Estimate initial $\omega$ using EM (with smart initialization for $\mathbf{k}$)
3: for $sweep = 1 - 5$ do
4:     Update $\mathbf{x}$ given current $\omega$ using graph cut
5:     Update $\omega$ given current $\mathbf{x}$ using EM
6: end for
User Edit: Update user trimap $T = \{T_F, T_B, T_U\}$ and go to step 3
```

Figure 3: The pseudocode for GrabCut+ with bounding box or lasso input.

the whole optimization procedure. The trimap T defines uniquely the segmentation x, which is used to initialize the unknown color model ω. This initialization step was not discussed in [36], however, it is quite important. One choice is a random initialization for $\mathbf{k}$, however, the following "smart initialization" works better. Consider the set of background pixels, $x_n = 0$. The first principal axis is computed from image data z_n of this set. Then the data is projected onto this axis and sorted accordingly. Finally, the sorted set is divided into K groups, which defines for each pixel n the assignment variable k_n. Given $\mathbf{k}$, a standard EM-style procedure for GMM fitting can be invoked. This means that in the "M" step, the Gaussian, i.e. π, μ, Σ, are fitted in a standard way, see [36] for details. In the "E" step $\mathbf{k}$ is optimized by enumerating all components $k \in K$, and choosing the one with lowest energy. In practice these two steps are executed 4 times. The foreground pixels are processed similarly.

The main procedure alternates the following two steps: i) Given ω, the segmentation $\mathbf{x}$ is inferred with graph cut as in sec. 2, ii) Given the segmentation $\mathbf{x}$, the unknown model ω is inferred using the above EM-style procedure for GMM fitting. In each step the total energy E is guaranteed to not increase. The method can be run until a local minimum is found, but in practice we simple stop it after a fixed number of 5 iterations. Fig. 2 shows the power of running the iterative GrabCut+ procedure. Finally, the user can update the trimap and the main procedure is run again. As in the previous section, re-using of flow gives a speed-up.

Note that a small modification of this procedure let us apply GrabCut+ for a standard brush interface [9], as done in the experiments. For that, step 1 in fig. 3 is initialized as $x_n = 0$ for $n \in T_B$, $x_n = 1$ for $n \in T_F$ and $x_n =$'?' for $n \in T_U$, where label '?' means unlabeled and those pixels are ignored in step 2 of the algorithm. In the extreme case where T_F is an empty set, we can proceed as in GrabCut+ and set $x_n = 1$ for $n \in T_U$. In the same way, it can be set $x_n = 0$ for $n \in T_U$ if $T_B = \emptyset$. Finally, it may occur that the user cannot provide any useful input, i.e. both sets T_F and T_B are empty. For example, the foreground object touches all for sides of the image, and hence the tightest bounding box is of the same size as the image. Another scenario is an automatic segmentation system, where the user might want to extract an object without any user input. An example is the web-based retrieval systems for vases [26]. In those cases, where $T_F = \emptyset$ and $T_B = \emptyset$, a simple procedure is run which places a foreground ellipse in the center of the image, where the area of the ellipse is half of the image area. The implicit assumption is that the object is more likely present in the image center. Note that a similar procedure is utilized in [26]. Note that for all of the above missing-data cases, i.e. $T_F = \emptyset$ and/or $T_B = \emptyset$, the tricks for better initialization, discussed in sec. 4.1 can potentially be used.

3.3 Properties of the GrabCut model

In the GrabCut model there is the freedom to choose an appropriate foreground and background color distributions ω. It is straightforward to see that distributions which are more compact, and model the data well, give a lower energy[5]. This means that implicitly it is assumed that both foreground and background segments are more likely represented by a compact distribution in color space.

One important question is whether the GrabCut model has an implicit bias towards certain segmentations. This was analyzed in Vicente et al. [44] by disregarding the smoothing term V in (3). They first showed that by using a histogram representation in RGB space it is possible to write the term "$\min_{\omega} E(\mathbf{x}, \omega, \mathbf{z})$" in (5) explicitly in the form of a new energy $E'(\mathbf{x}, \mathbf{z})$ with higher-order cliques on $\mathbf{x}$. This higher-order energy E' has two different types of terms: i) a convex function over $\sum_n x_n$, and ii) a concave function, for each histogram bin k, over $\sum_n x_n [n \in Bin(k)]$, i.e. all pixels which are assigned to bin k. The convex function (first type) has lowest energy if exactly half of the pixels in the image are assigned to foreground and background respectively. Hence, this is a bias towards balanced segmentations. The concave function (second type) has lowest energy if all pixels, which are assigned to the same bin, have the same label, either 0 or 1. Note that these two types of terms often counter-balance each other in practice so that the optimal segmentation is not a degenerate solution, e.g. not all undefined pixels are either all foreground or all background. In an extreme case the bias towards balanced segmentation is more prominent. This is when all pixels are assigned to unique histogram bins. Then all concave terms are constants, so that the energy consists of the convex part only. In the other extreme case, when all pixels are assigned to the same histogram bin, the bias disappears however, since then concave and convex terms cancel each other out. Note, an interesting observation was made in [44] that results do improve considerable when choosing the weight of this bias individually for each image.

Unfortunately, optimizing the higher-order energy E', with respect to $\mathbf{x}$, is an NP-hard problem [44]. An optimization procedure for E' was suggested in [44], based on dual decomposition ([3] and chapter 22, which also provides a lower bound for the energy. This procedure achieved global optimality in 61% of test cases for the GrabCut database of 49 images [18] and a bounding box input. However, for the remaining 39% of test cases, the dual decomposition approach performed rather poorly. Many of those test cases were "camouflage images", where foreground and background colors

[5]For example, assume all foreground pixels have the same color then the lowest unary term is achieved by modeling the foreground distribution with one (or many identical) Gaussian with minimum variance.

do overlap considerably, (example in fig. 2 in [44]). In [44] a comparison with an iterative GrabCut-style procedure, as described in fig. 3, was done, with the difference that histograms replace GMMs and that the whole image outside the box is used as background training data. The iterative procedure achieved global optimality only in 4% of cases, i.e. very often got trapped in a local minimum. Despite this disadvantage, the iterative procedure performed well in terms of error rate. In 80% of test cases, where dual-decomposition achieved lower energy, the iterative procedure is only slightly worse in terms of error rate. However, for the remaining 20% of cases the iterative procedure is often considerably better. Hence, the iterative procedure achieved even a lower total error rate (8.1%) than the dual-decomposition approach (10.5%). It can be expected that the iterative procedure will perform even better with a different initialization step, as discussed in sec. 4.1. Note that [44] also suggested a semi-global procedure, which eventually beats the iterative process for all test cases.

4 Background Removal: Image Segmentation in MS Office 2010

Designing a product, based on the GrabCut+ tool (sec. 3) means that many interesting, practical and theoretical, issues have to be addressed. This section discusses all of the topics for which the Background Removal tool and the GrabCut+ tool differ. We begin by revisiting the optimization procedure and then examine additional model constraints.

4.1 Initialization and the Bounding Box Prior

It turns out that choosing the initial segmentation $\mathbf{x}$, based on the user trimap T, i.e. step 1 in fig. 3, is crucial for the performance of the full procedure. In the following three different initialization schemes are compared for the bounding box input. All approaches have in common that $x_n = 0$ for $n \in T_B$, however, they differ in the treatment of pixels in T_U. The first approach, called *"InitFullBox"*, is described in sec. 3.2, and sets $x_n = 1$ for all $n \in T_U$. The second method, called *"InitThirds"*, was suggested in [28]. First, a background color model is trained from pixels in T_B. Then the probability of all pixels in T_U are evaluated under the background GMM. One third of the pixels with lowest probability are set to foreground, and one third with highest probability are set to background[6]. The remaining pixels are set to '?', i.e. are ignored in step 2 of fig. 3. The last approach, called *"InitParametric"*, is implemented in the Background Removal tool and is similar to the "InitThirds" procedure but additionally considers the smoothing term V of the energy. For this a new parametric energy is introduced $E''(\mathbf{x},\omega,\mathbf{z}) = E(\mathbf{x},\omega,\mathbf{z}) + \sum_n \lambda x_n$, with E as defined in sec. 3.1. Here ω is chosen such that the background GMM is trained from pixels in T_B, and the distribution for the foreground is a constant, uniform distribution. The global optimum of E'' for all continuous values of λ can be computed efficiently using parametric maxflow [23][7].

[6]The choice of using a third as the threshold is arbitrary and should be learned from data.

[7]Since we were not aware of parametric maxflow in 2004, a simple iterative procedure is utilized, which re-uses flow.

From the set of all solutions for E'', one solution $\mathbf{x}$ is selected using the following heuristic. The segmentation with smallest foreground area is selected which also meets the following criteria: The maximum distance of the largest connected component to any four sides of the box is smaller than a fixed threshold (e.g. 25% of a side of the bounding box)[8]. The inspiration for the "InitParametric" initialization procedure is the following. First, the user is more likely to select a bounding box which is tight around the object. We refer to this idea as the *bounding box prior* and it motivated the work in [28]. Secondly, as in "InitThirds" the foreground segmentation is often "far away" in feature space from the given background distribution. However, in contrast to "InitThirds" the foreground segmentation is spatially coherent with this procedure.

An experiment gives an indication of the quality of results which can be achieved with these three methods. For this test the GrabCut data-set (50 images) is used together with the bounding boxes from [28] (see online [18])[9]. It has to be stressed that the following error rates have to be taken with care, since the data-set is rather small[10], and parameters were not trained[11]. The exact settings for GrabCut+ are as defined in sec. 3, i.e. $\lambda_1 = 5, \lambda_2 = 50, K = 7$. Results are as follows. Initializing GrabCut+ with "InitFullBox" gave an error rate of 9.0%.[12] It seems that the initialization of "InitThirds" is clearly a better choice since the error rate dropped from 9.0% to 5.0%, which is the same conclusion as in [28]. Running the Background Removal tool, which uses "InitParametric", gave a slightly higher error rate of 5.95%.

Note that this experiment did not enforce the very sensible bounding box prior, which ensures that the final segmentation is close to the user selected bounding box. Indeed, by enforcing this prior the error rate can be reduced to 3.7%. The algorithm to achieve this is described in [28] and runs graphcut iteratively while forcing certain pixels to belong to foreground. We refer the interested reader to [28] for several alternative procedures for enforcing the bounding box prior. An interesting direction of future work could be to use as initialization procedure "InitThirds" and to exploit the parametric maxflow approach of "InitParametric" to enforce the bounding box prior.

4.2 Modeling User Intention

The ultimate goal of a segmentation system is that a user achieves the desired result in as short time as possible. For this goal, the energy defined in sec. 3.1 might not be the optimal model. One problem with the above model is that it is agnostic to the *sequence* of user interactions. By exploiting this sequence the intention of the user can

[8]Note that this is the same criteria as the weak tightness condition defined in [28].

[9]Note, the bounding boxes from [28] deviate slightly from the original set of bounding boxes. The bounding boxes which are different touch the image boundary, while the object does not. This modification simplifies the segmentation task and also removes the problem that in one image the original bounding box was of the same size as the image itself.

[10]A larger, e.g. 1000+, data-set with high-quality ground truth is needed in the future. Note that for product testing a medium-sized data-set was created, which also includes images which are not photographs, e.g. graphics and hand-drawn sketches.

[11]As discussed in detail later, image segmentation is an interactive process hence parameters have to be trained anyway with the user in the loop, as in sec. 5.2.

[12]One small modification did reduce the error rate to 7.1%. This was done by choosing a random initialization for $\mathbf{k}$ in step 2 in fig. 3. It shows that initialization is very important and can effect the final result considerably, and that the data-set may be to small.

be modeled in a better way. Two simple ideas, which are realized in the Background Removal tool, will be discussed[13]. Other ideas for modeling the user's intention are presented in recent work [45, 30], which investigate alternative models and different user interfaces.

The first idea is to avoid the so-called "fluctuation effect" [30]. Consider a current imperfect segmentation where the user has placed the (latest) foreground brush stroke. Two effects are undesirable: a) pixels change label from foreground to background, and b) pixels which are spatially far away, change label from background to foreground, since the user may not notice it. We enforce the sensible constraint that the change in the segmentation must be, in this case, from background to foreground and also 4-connected to the latest foreground brush. Achieving connectivity is in general an NP-hard problem (see chapter 22and [43]), hence we solve it with a simple post-processing step[14]. The same procedure is applied for a background brush. It is worth to note that with this connectivity prior, parameters in the GrabCut+ model may be chosen quite differently, see [34][15]. Also, note that many systems which do not use an explicit unary term, such as [2] and random walker [16], are guaranteed to satisfy this connectivity property[16].

The second idea, achieves the desired property that the latest brush stroke always has a noticeable effect on the current segmentation, which is related to the "progressive labeling" concept in [30]. Consider the case where the current segmentation has a dominant color of red in the background, and a small region of the true foreground is also red, which is, however, currently incorrectly labeled. A foreground brush in this small region may fail to select the whole region, since the unary terms in the region strongly favor the background label, i.e. red being background. The underlying problem is that the general global color model as defined in sec. 3.1 is not always appropriate for modeling objects. A practical solution to overcome this problem is to simply give pixels which are in the latest foreground brush stroke a much higher weight, e.g. 80%, and all other foreground pixels a lower weight (e.g. 20% weight). The same procedure is applied for a (latest) background brush[17]. Note that this aggressive color modeling procedure does only work in conjunction with the above idea of connectivity with the latest brush stroke.

[13]These ideas were realized in Microsoft Expression Acrylic Graphics Designer (technology preview). Unfortunately, due to constraints in the Microsoft Office product, it was not possible to realize them exactly, as described below. The difference is that in the MS Office version all user foreground and background brush strokes are treated as one (latest) foreground and background brush stroke respectively.

[14]Note that other automatic techniques can be used in the future, such as [35] based on [43], or the the "geodesic star convexity prior" of [17].

[15]The reason is that one of the main effects of the Ising prior in an MRF (weight λ_1 in (3)) is to smooth out wrongly labeled isolated regions. By enforcing connectivity these isolated regions are not permitted in the solution space, hence a different weight for the Ising prior might perform better.

[16]In this context, we also post-process the initial segmentation, which is the result of a bounding box (or lasso) input, such that only one 4-connected foreground component is present. Note that no constraint on the background label is enforced, since many objects do have holes.

[17]An improvement could be to model, in case of a foreground brush, the background color model in a different way, such that it is more representative for this segmentation task.

4.3 Dealing with Large Images and Semi-Transparency

For the segmentation of large images, e.g. 20MPixels, it is computationally too expensive to perform graph cut in the given resolution. In the Background Removal tool a 3-level multi-resolution approach is adapted. The full segmentation procedure is run only on a small resolution of about 0.07MPixels[18]. Then the segmentation result is up-scaled to a medium resolution of about 0.8MPixels[19], where graph cut is run again in a small band around the up-scaled segmentation, as in [31]. An important observation, which is not discussed in [31], is that the energy for different resolutions must be learned (or set) differently. This can be seen from the fact that ratio of segmentation area over boundary length is resolution-dependent. In [21] it is shown how to set parameters, in particular λ_1, λ_2 and β in (3), for a smaller resolution given that the energy was learned on a higher resolution. Finally, the medium-level solution is up-scaled to the final resolution with a fast and simple procedure, which does not look at the input image. This is done by up-scaling the binary segmentation via a 2-pass re-sampling operation with a Gaussian kernel. This produces a grey-scale image which we threshold at the value of 0.5 to yield a final binary high-resolution segmentation. An alternative approach is the joint bilateral up-scaling technique of [24] as used in [30]. One practical aspect is that the values for small and medium resolution have to be taken with care since it can happen that the computation time for graph cut in the band for the medium resolution is considerable high. Another practical aspect is that the user brushes are given in the highest resolution. Hence, care must be taken that the final segmentation obeys the user constraints.

Going from a binary segmentation to a continuous, grey-level segmentation, is the topic of image matting and a large research area in itself, see e.g. [46, 35]. In the Background Removal tool we deal with it in a fast way which is practically satisfying in many cases. The image of an object boundary is always slightly blurred, due to camera effects (discrete CCD chip and point-spread-function of camera)[20]. We model this by simply blurring the hard segmentation with a Gaussian kernel, where the kernel size is image-resolution dependent. The true foreground colors are then determined using the color-stealing process of [36].

5 Evaluation and Future Work

As mentioned in the introduction, the question of what is the best interactive segmentation system today is hard to answer. To cast some light on this question, two specific experiments were performed. First, with a static user input. Second, with a so-called “robot user” which simulates a simple novice user. The robot user is used to train and compare different systems in a truly interactive setting.

[18]The image is scaled-down such that the longest image-side is a maximum of 300 pixels long.

[19]If the longest image-side is more than 1280pixels, it is scaled-down to 1024pixels.

[20]Effects like true transparency, sub-pixel structure (e.g. hair), or out-of-focus background, are ignored.

5.1 Static User Input

A plausible type of user interaction, for objects with not excessively long boundaries, is that the user draws with a “fat pen” around the boundary of the object, which produces a relatively tight trimap. Obviously for such a user input a method should exploit the fact that the true boundary is more likely in the middle of the user drawn band. As above, the GrabCut database (50 images) is used which provides such a trimap, derived by simply eroding the ground truth segmentation, see online [18]. Several articles have reported error statistics for this data-set, using the percentage of misclassified pixels within the unknown trimap region T_U[21]. As above, the following error rates have to be taken with care since the database is small and the methods were not properly trained. Also, all parameter settings for GrabCut+ (and variants of it) are as described above.

Applying simple graph cut without any global color modeling, i.e. energy in (1) without unary term U, gives an error rate of 9.0%. As discussed in detail in chapter 8, one bias of graph cut is the so-called “shrinking bias”, i.e. segmentations with a short boundary are preferred. In contrast, Random walker [16] has less of a “shrinking bias” and instead a “proximity bias” towards segmentations which are equally far away from the given foreground and background trimap respectively. Obviously, this is a better bias for this data-set, hence the error rate for random walker is 5.4% (see [15])[22]. Note, in [40] (see also chapter 8) a continuum of solutions is presented which vary with respect to the proximity bias. As to be expected, the setting ($p = \infty$ in [40]) which exploit the proximity bias most, is the best. On this note, a simple baseline method which ignores the image data achieves a quite low error rate of 4.5%. The baseline method simply classifies each pixel according to the Euclidian distance to the foreground and background region respectively[23]. It is worth to note that there is a method which beats the baseline, that is Random walker with an “adaptive thresholding”, which better exploits the proximity bias, see [15].

Finally, the effect of adding global color models is investigated. The error rate of graph cut reduces from 9.0% to 6.6% using the GrabCut+ algorithm in fig. 3 with one sweep (step 3). Multiple iterations of GrabCut+ reduce the error rate further to 5.6%, and the Background Removal tool achieves basically the same error rate of 5.8%.[24] To conclude, global color models do help considerably graph cut-based techniques, and they may also help the best perform methods for this data-set, as also conjectured in [15].

[21]A small fraction of pixels in the ground truth are unlabeled, due to transparency effects. These pixels are not counted when computing the error rate.

[22]In [15] some variations of the random walker formulation are given, which however, produce quantitatively the same results.

[23]The baseline method dose not treat thin structures well, hence a skeletonisation approach might even perform better.

[24]Note that by exploiting additionally the idea that the relevant training data for the background GMM is more likely in a strip around the unknown region (as provided with the data-set), the error rate of GrabCut+ drops from 5.6% to 5.3%.

5.2 Dynamic user input

The set-up is described in detail in Gulshan et. al. [17], hence only some aspects are mentioned here. To measure the amount of user interaction in an interactive system, we invented the so-called "robot user" [34]. It can be used for both learning and evaluating segmentation systems, see [34]. Given a new image, it starts with an initial set of brush strokes[25] and computes a segmentation. It then places a circular brush stroke in the largest connected component of the segmentation *error* area, placed at a point farthest from the boundary of the component. The process is repeated up to 20 times, generating a sequence of 20 simulated user strokes, which is different for each algorithm. Fig. 4(b) shows an example (see also video in [18]). From the sequence of interactions, one number for the amount of user interaction is derived which measures, in rough words, the average number of brush strokes necessary to achieve a good quality result (details in [17]). A small user study confirmed that the interaction effort of the robot user, indeed, correlates reasonable well with the true effort of a *novice user*, see [34].

The dataset for this experiment consists of 151 images with ground truth segmentations, which is a mix of existing datasets including GrabCut and VOC'09, see [18]. The free parameters for all systems were trained using cross-validation, 75 – 76 split repeated 10 times.

Table 1 presents the results for five different methods, and fig. 4(d) depicts the average error rate for the sequence of robot user interactions. The "Geo" method of [2], based on geodesic distance, performed worst. This is not surprising since the method does not regularize the boundary and is sensitive to the exact location of brush strokes (see chapter 8). Fig. 4(c) gives an example. Graph cut "GC" (sec. 2) performed considerably better[26]. The main difference of graph cut compared to the following three systems is that it does not impose any "shape" prior. Random Walker "RW" [16], for instance, guarantees a connectivity of the segmentation with respect to brushes, e.g. a pixel with label 0 is 4/8-connected to a background brush. Hence, even without global color models Random Walker performs as well as graph cut. The Background Removal tool "BR" is the second best performing system (example in fig. 4(b)). It exploits the same connectivity property as Random Walker, but additionally utilizes global color models. Finally, the best performing method "GSC" [17] imposes a strong "geodesic star convexity prior" on top of the simple graph cut system "GC". Note that this convexity prior is even more restrictive than a connectivity prior (see fig. 14 in [17]), which seems to be an advantage in practice[27].

5.3 Future Work

Many ideas for future work were already mentioned above. Certainly, the model for Background Removal can be improved further using stronger shape priors, e.g. [17], improved local MRF modeling, e.g. flux [35], or better global color models, e.g. [17].

[25]They were chosen manually with one stroke for foreground and three strokes for background.

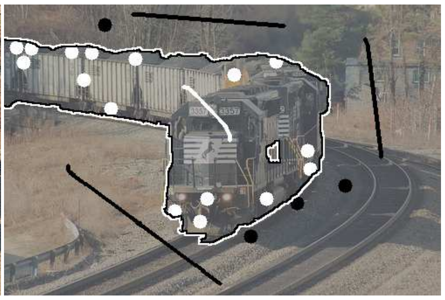
[26]Here the implementation of [18] was used. It includes an important, additional trick, which mixes the GMMs with a uniform distribution.

[27]This is supported by testing against another variant of "GC", which performed slightly worse (effort 10.66). It removes all foreground islands which are not connected to a foreground brush stroke in a post-processing step.

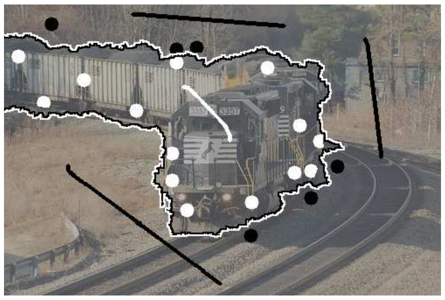
Method	Geo [2]	GC (sec. 2)	RW [16]	BR (sec. 3)	GSC [17]
Effort	15.14	12.35	12.31	10.82	9.63

Table 1: Comparison of five different systems in terms of the average number of brush strokes needed by the robot user to achieve good results.

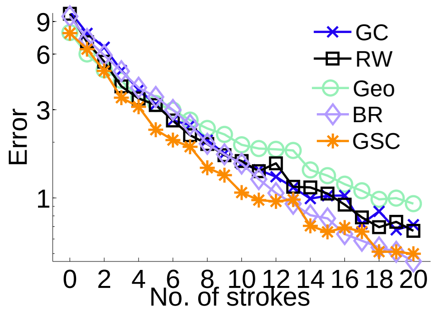
(a)



(b)



(c)



(d)

Figure 4: (a) Input image (original in color). (b) Result of the robot user using the "BR" system, with 0.85% of misclassified pixels (image colors adapted for better visualization). The segmentation is outlined with a black-white line, and the robot user inputs are white and black circles for the foreground and background respectively (long strokes are initial, manual user strokes). (c) Result from "Geo" system which is considerably worse, error 1.61%. (d) Performance of five different systems utilizing the robot user (error in log-scale).

Apart from improving the model we believe that further improvements may be achieved by focusing more on user aspects.

6 Acknowledgement

Many people helped with discussions, experiments and great new ideas on the topic of interactive image segmentation. We would like gratefully thank, Toby Sharp, Varun Gulshan, Victor Lempitsky, Pushmeet Kohli, Sara Vicente, Antonio Criminisi, Christoph Rhemann, Dheeraj Singaraju, Hannes Nickisch, and Patrick Perez.

References

[1] Adobe Systems Incorp. Adobe Photoshop User Guide. 2002.

[2] X. Bai and G. Sapiro. Geodesic matting: A framework for fast interactive image and video segmentation and matting. In *Int. J. Computer Vision*, 2009.

[3] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[4] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *ICCV*, 1999.

[5] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[6] A. Blake, P. Kohli, and C. Rother. *Advances in Markov Random Fields for Vision and Image Processing*. MIT Press, 2011 (to appear).

[7] A. Blake, C. Rother, and P. Anandan. *Foreground extraction using iterated graph cuts*. United States Patent 7660463 (Filing Date 2004), 2009.

[8] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive Image Segmentation using an adaptive GMMRF model. In *Proc. European Conf. Computer Vision*, 2004.

[9] Y. Boykov and M-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *Proc. IEEE Int. Conf. on Computer Vision*, 2001.

[10] Y. Boykov and V. Kolmogorov. Computing Geodesics and Minimal Surfaces via Graph Cut. In *Proc. IEEE Int. Conf. on Computer Vision*, 2003.

[11] Y. Boykov and V. Kolmogrov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, volume 26, pages 1124–1137, 2004.

[12] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. In *Proc. IEEE Int. Conf. on Computer Vision*, 1995.

[13] T. Chan and L. Vese. Active contours without edges. *IEEE Trans. Image Processing*, 10(2), 2001.

[14] I. J. Cox, S. B. Rao, and Y. Zhong. Ratio regions: a technique for image segmentation. In *ICPR*, 1996.

[15] O. Duchenne, J.-Y. Audibert, R. Keriven, J. Ponce, and F. Segonne. Segmentation by transduction. In *Proc. IEEE Conf. Computer Vision and Pattern Recog.*, 2008.

[16] L. Grady. Random walks for image segmentation. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, volume 28, pages 1768–1783, 2006.

[17] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman. Geodesic star convexity for interactive image segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recog.*, 2010.

[18] http://research.microsoft.com/en-us/projects/i3l/segmentation.aspx.

[19] I. Jermyn and H. Ishikawa. Globally optimal regions and boundaries as minimum ratio weight cycles. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, volume 23, 2001.

[20] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proc. IEEE Int. Conf. on Computer Vision*, pages 259–268, 1987.

[21] P. Kohli, V. Lempitsky, and C. Rother. Uncertainty driven multi-scale energy minimization. In *Deutsche Arbeitsgemeinschaft fuer Mustererkennung (DAGM)*, 2010.

[22] V. Kolmogorov and Y. Boykov. What metrics can be approximated by geo-cuts, or global optimization of length/area and flux. In *Proc. IEEE Int. Conf. on Computer Vision*, 2005.

[23] V. Kolmogorov, Y. Boykov, and C. Rother. Applications of parametric maxflow in computer vision. In *Proc. IEEE Int. Conf. on Computer Vision*, 2007.

[24] J. Kopf, M. Cohen, D. Lischinski, and M. Uyttendaele. Joint bilateral upsampling. *Proc. ACM Siggraph*, 2007.

[25] M. P. Kumar, P.H.S. Torr, and A. Zisserman. Obj cut. In *Proc. IEEE Conf. Computer Vision and Pattern Recog.*, 2005.

[26] D. Kurtz, G. Parker, D. Shotton, G. Klyne, F. Schroff, A. Zisserman, and Y. Wilks. Claros - bringing classical art to a global public. In *Fifth IEEE International Conference on e-Science*, 2009.

[27] V. Lempitsky, A. Blake, and C. Rother. Image segmentation by branch-and-mincut. In *Proc. European Conf. Computer Vision*, 2008.

[28] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. Image segmentation with a bounding box prior. In *Proc. IEEE Int. Conf. on Computer Vision*, 2009.

[29] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *Proc. ACM Siggraph*, 2004.

[30] J. Liu, J. Sun, and H.-Y. Shum. Paint selection. *Proc. ACM Siggraph*, 2009.

[31] H. Lombaert, Y. Sun, L. Grady, and C. Xu. A multilevel banded graph cuts method for fast image segmentation. In *Proc. IEEE Int. Conf. on Computer Vision*, 2005.

[32] E.N. Mortensen and W.A. Barrett. Intelligent scissors for image composition. *Proc. ACM Siggraph*, pages 191–198, 1995.

[33] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Comm. Pure Appl. Math.*, 42:577685, 1989.

[34] H. Nickisch, P. Kohli, and C. Rother. Learning an interactive segmentation system. In *Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP)*, 2010.

[35] C. Rhemann, C. Rother, P. Kohli, and M. Gelautz. A spatially varying psf-based prior for alpha matting. In *Proc. IEEE Conf. Computer Vision and Pattern Recog.*, 2010.

[36] C. Rother, V. Kolmogorov, and A. Blake. Grabcut — interactive foreground extraction using iterative graph cuts. *Proc. ACM Siggraph*, 2004.

[37] M.A. Ruzon and C. Tomasi. Alpha estimation in natural images. In *Proc. IEEE Conf. Comp. Vision and Pattern Recog.*, 2000.

[38] J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, volume 22, 2000.

[39] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Proc. European Conf. Computer Vision*, 2006.

[40] D. Singaraju, L. Grady, and R. Vidal. P-brush: Continuous valued mrfs with normed pairwise distributions for image segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recog.*, 2009.

[41] M. Szummer, P. Kohli, and D. Hoiem. Learning crfs using graph cuts. In *Proc. European Conf. Computer Vision*, 2008.

[42] M. Unger, T. Pock, D. Cremers, and H. Bishop. Tvseg - interactive total variation based image segmentation. In *Proc. British Machine Vision Conf.*, 2008.

[43] S. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. In *Proc. IEEE Conf. Computer Vision and Pattern Recog.*, 2008.

[44] S. Vicente, V. Kolmogorov, and C. Rother. Joint optimization of segmentation and appearance models. In *Proc. IEEE Int. Conf. on Computer Vision*, 2009.

[45] C. Wang, Q. Yang, M. Chen, X. Tang, and Z. Ye. Progressive cut. In *International Multimedia Conference*, 2006.

[46] J. Wang and M. Cohen. Image and video matting: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(2), 2007.