"GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts

Carsten Rother*   Vladimir Kolmogorov†   Andrew Blake‡

Microsoft Research Cambridge, UK

Figure 1: **Three examples of GrabCut .** The user drags a rectangle loosely around an object. The object is then extracted automatically.

Abstract

The problem of efficient, interactive foreground/background segmentation in still images is of great practical importance in image editing. Classical image segmentation tools use either texture (colour) information, e.g. Magic Wand, or edge (contrast) information, e.g. Intelligent Scissors. Recently, an approach based on optimization by graph-cut has been developed which successfully combines both types of information. In this paper we extend the graph-cut approach in three respects. First, we have developed a more powerful, iterative version of the optimisation. Secondly, the power of the iterative algorithm is used to simplify substantially the user interaction needed for a given quality of result. Thirdly, a robust algorithm for "border matting" has been developed to estimate simultaneously the alpha-matte around an object boundary and the colours of foreground pixels. We show that for moderately difficult examples the proposed method outperforms competitive tools.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.4.6 [Image Processing and Computer Vision]: Segmentation—Pixel classification; partitioning

Keywords: Interactive Image Segmentation, Graph Cuts, Image Editing, Foreground extraction, Alpha Matting

1   Introduction

This paper addresses the problem of efficient, interactive extraction of a foreground object in a complex environment whose background cannot be trivially subtracted. The resulting foreground object is an alpha-matte which reflects the proportion of foreground and background. The aim is to achieve high performance at the cost of only modest interactive effort on the part of the user. High performance in this task includes: accurate segmentation of object from background; subjectively convincing alpha values, in response to blur, mixed pixels and transparency; clean foreground colour,

*e-mail: carrot@microsoft.com

†e-mail: vnk@microsoft.com

‡e-mail: ablake@microsoft.com

free of colour bleeding from the source background. In general, degrees of interactive effort range from editing individual pixels, at the labour-intensive extreme, to merely touching foreground and/or background in a few locations.

1.1   Previous approaches to interactive matting

In the following we describe briefly and compare several state of the art interactive tools for segmentation: Magic Wand, Intelligent Scissors, Graph Cut and Level Sets and for matting: Bayes Matting and Knockout. Fig. 2 shows their results on a matting task, together with degree of user interaction required to achieve those results.

Magic Wand   starts with a user-specified point or region to compute a region of connected pixels such that all the selected pixels fall within some adjustable tolerance of the colour statistics of the specified region. While the user interface is straightforward, finding the correct tolerance level is often cumbersome and sometimes impossible. Fig. 2a shows the result using Magic Wand from Adobe Photoshop 7 [Adobe Systems Incorp. 2002]. Because the distribution in colour space of foreground and background pixels have a considerable overlap, a satisfactory segmentation is not achieved.

Intelligent Scissors   (a.k.a.   Live Wire or Magnetic Lasso) [Mortensen and Barrett 1995] 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. 2b the Magnetic Lasso of Photoshop 7 was used. The main limitation of this tool is apparent: for highly texture (or un-textured) regions many alternative "minimal" paths exist. Therefore many user interactions (here 19) were necessary to obtain a satisfactory result. Snakes or Active Contours are a related approach for automatic refinement of a lasso [Kass et al. 1987].

Bayes matting   models colour distributions probabilistically to achieve full alpha mattes [Chuang et al. 2001] which is based on [Ruzon and Tomasi 2000]. The user specifies a "trimap" $T = \{T_B, T_U, T_F\}$ in which background and foreground regions T_B and T_F are marked, and alpha values are computed over the remaining region T_U. High quality mattes can often be obtained (fig. 2c), but only when the T_U region is not too large and the background/foreground colour distributions are sufficiently well separated. A considerable degree of user interaction is required to construct an internal and an external path.

Knockout 2   [Corel Corporation 2002] is a proprietary plug-in for Photoshop which is driven from a user-defined trimap, like Bayes matting, and its results are sometimes similar (fig. 2d), sometimes of less quality according to [Chuang et al. 2001].

309

Graph Cut [Boykov and Jolly 2001; Greig et al. 1989] is a powerful optimisation technique that can be used in a setting similar to Bayes Matting, including trimaps and probabilistic colour models, to achieve robust segmentation even in camouflage, when foreground and background colour distributions are not well separated. The system is explained in detail in section 2. Graph Cut techniques can also be used for image synthesis, like in [Kwatra et al. 2003] where a cut corresponds to the optimal smooth seam between two images, e.g. source and target image.

Level sets [Caselles et al. 1995] is a standard approach to image and texture segmentation. It is a method for front propagation by solving a corresponding partial differential equation, and is often used as an energy minimization tool. Its advantage is that almost any energy can be used. However, it computes only a local minimum which may depend on initialization. Therefore, in cases where the energy function can be minimized exactly via graph cuts, the latter method should be preferable. One such case was identified by [Boykov and Kolmogorov 2003] for computing geodesics and minimal surfaces in Riemannian space.

1.2 Proposed system: GrabCut

Ideally, a matting tool should be able to produce continuous alpha values over the entire inference region T_U of the trimap, without any hard constraint that alpha values may only be 0 or 1. In that way, problems involving smoke, hair, trees etc., could be dealt with appropriately in an automatic way. However, in our experience, techniques designed for solving that general matting problem [Ruzon and Tomasi 2000; Chuang et al. 2001] are effective when there is sufficient separation of foreground and background color distributions but tend to fail in camouflage. Indeed it may even be that the general matting problem is not solvable in camouflage, in the sense that humans would find it hard to perceive the full matte. This motivates our study of a somewhat less ambitious but more achievable form of the problem.

First we obtain a "hard" segmentation (sections 2 and 3) using iterative graph cut . This is followed by border matting (section 4) in which alpha values are computed in a narrow strip around the hard segmentation boundary. Finally, full transparency, other than at the border, is not dealt with by GrabCut . It could be achieved however using the matting brush of [Chuang et al. 2001] and, in our experience, this works well in areas that are sufficiently free of camouflage.

The novelty of our approach lies first in the handling of segmentation. We have made two enhancements to the graph cuts mechanism: "iterative estimation" and "incomplete labelling" which together allow a considerably reduced degree of user interaction for a given quality of result (fig. 2f). This allows GrabCut to put a light load on the user, whose interaction consists simply of dragging a rectangle around the desired object. In doing so, the user is indicating a region of background, and is free of any need to mark a foreground region. Secondly we have developed a new mechanism for alpha computation, used for border matting, whereby alpha values are regularised to reduce visible artefacts.

2 Image segmentation by graph cut

First, the segmentation approach of Boykov and Jolly , the foundation on which GrabCut is built, is described in some detail.

2.1 Image segmentation

Their paper [Boykov and Jolly 2001] addresses the segmentation of a monochrome image, given an initial trimap T. The image is an array $\mathbf{z} = (z_1, \ldots, z_n, \ldots, z_N)$ of grey values, indexed by the (single) index n. The segmentation of the image is expressed as an array of "opacity" values $\underline{\alpha} = (\alpha_1, \ldots, \alpha_N)$ at each pixel. Generally $0 \le \alpha_n \le 1$, but for hard segmentation $\alpha_n \in \{0,1\}$, with 0 for background and 1 for foreground. The parameters $\underline{\theta}$ describe image foreground and background grey-level distributions, and consist of histograms of grey values:

$$\underline{\theta} = \{h(z; \alpha), \alpha = 0, 1\}, \tag{1}$$

one for background and one for foreground. The histograms are assembled directly from labelled pixels from the respective trimap regions T_B, T_F. (Histograms are normalised to sum to 1 over the grey-level range: $\int_z h(z; \alpha) = 1$.)

The segmentation task is to infer the unknown opacity variables $\underline{\alpha}$ from the given image data $\mathbf{z}$ and the model $\underline{\theta}$.

2.2 Segmentation by energy minimisation

An energy function $\mathbf{E}$ is defined so that its minimum should correspond to a good segmentation, in the sense that it is guided both by the observed foreground and background grey-level histograms and that the opacity is "coherent", reflecting a tendency to solidity of objects. This is captured by a "Gibbs" energy of the form:

$$\mathbf{E}(\underline{\alpha}, \underline{\theta}, \mathbf{z}) = U(\underline{\alpha}, \underline{\theta}, \mathbf{z}) + V(\underline{\alpha}, \mathbf{z}) . \tag{2}$$

The data term U evaluates the fit of the opacity distribution $\underline{\alpha}$ to the data $\mathbf{z}$, given the histogram model $\underline{\theta}$, and is defined to be:

$$U(\underline{\alpha}, \underline{\theta}, \mathbf{z}) = \sum_n -\log h(z_n; \alpha_n) . \tag{3}$$

The smoothness term can be written as

$$V(\underline{\alpha}, \mathbf{z}) = \gamma \sum_{(m,n) \in \mathbf{C}} dis(m,n)^{-1} \, [\alpha_n \ne \alpha_m] \exp{-\beta (z_m - z_n)^2}, \tag{4}$$

where $[\phi]$ denotes the indicator function taking values $0, 1$ for a predicate ϕ, $\mathbf{C}$ is the set of pairs of neighboring pixels, and where $dis(\cdot)$ is the Euclidean distance of neighbouring pixels. This energy encourages coherence in regions of similar grey-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). When the constant $\beta = 0$, the smoothness term is simply the well-known Ising prior, encouraging smoothness everywhere, to a degree determined by the constant γ. It has been shown however [Boykov and Jolly 2001] that it is far more effective to set $\beta > 0$ as this relaxes the tendency to smoothness in regions of high contrast. The constant β is chosen [Boykov and Jolly 2001] to be:

$$\beta = \left(2 \left\langle (z_m - z_n)^2 \right\rangle \right)^{-1}, \tag{5}$$

where $\langle \cdot \rangle$ denotes expectation over an image sample. This choice of β ensures that the exponential term in (4) switches appropriately between high and low contrast. The constant γ was obtained as 50 by optimizing performance against ground truth over a training set of 15 images. It proved to be a versatile setting for a wide variety of images (see [Blake et al. 2004]).

Now that the energy model is fully defined, the segmentation can be estimated as a global minimum:

$$\hat{\underline{\alpha}} = \arg\min_{\underline{\alpha}} \mathbf{E}(\underline{\alpha}, \underline{\theta}). \tag{6}$$

Minimisation is done using a standard minimum cut algorithm [Boykov and Jolly 2001; Kolmogorov and Zabih 2002]. This algorithm forms the foundation for hard segmentation, and the next section outlines three developments which have led to the new hard segmentation algorithm within GrabCut . First, the monochrome image model is replaced for colour by a Gaussian Mixture Model (GMM) in place of histograms. Secondly, the one-shot minimum cut estimation algorithm is replaced by a more powerful, iterative procedure that alternates between estimation and parameter learning. Thirdly, the demands on the interactive user are relaxed by allowing incomplete labelling — the user specifies only T_B for the trimap, and this can be done simply by placing a rectangle or a lasso around the object.

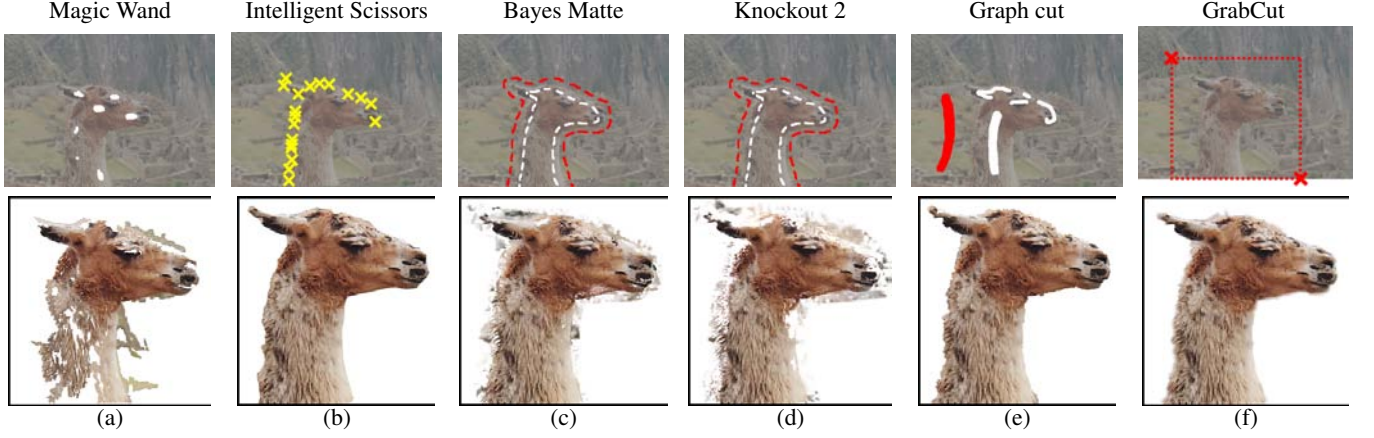| Magic Wand	Intelligent Scissors	Bayes Matte	Knockout 2	Graph cut	GrabCut
(a)	(b)	(c)	(d)	(e)	(f)

Figure 2: **Comparison of some matting and segmentation tools.** The top row shows the user interaction required to complete the segmentation or matting process: white brush/lasso (foreground), red brush/lasso (background), yellow crosses (boundary). The bottom row illustrates the resulting segmentation. GrabCut appears to outperform the other approaches both in terms of the simplicity of user input and the quality of results. Original images on the top row are displayed with reduced intensity to facilitate overlay; see fig. 1. for original. Note that our implementation of Graph Cut [Boykov and Jolly 2001] uses colour mixture models instead of grey value histograms.

3 The GrabCut segmentation algorithm

This section describes the novel parts of the GrabCut hard segmentation algorithm: iterative estimation and incomplete labelling.

3.1 Colour data modelling

The image is now taken to consist of pixels z_n in RGB colour space. As it is impractical to construct adequate colour space histograms, we follow a practice that is already used for soft segmentation [Ruzon and Tomasi 2000; Chuang et al. 2001] and use GMMs. Each GMM, one for the background and one for the foreground, is taken to be a full-covariance Gaussian mixture with K components (typically $K = 5$). 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 background or the foreground model, according as $\alpha_n = 0$ or 1[1].

The Gibbs energy (2) for segmentation now becomes

$$\mathbf{E}(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) = U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) + V(\underline{\alpha}, \mathbf{z}), \qquad (7)$$

depending also on the GMM component variables $\mathbf{k}$. The data term U is now defined, taking account of the colour GMM models, as

$$U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) = \sum_n D(\alpha_n, k_n, \underline{\theta}, z_n), \qquad (8)$$

where $D(\alpha_n, k_n, \underline{\theta}, z_n) = -\log p(z_n \mid \alpha_n, k_n, \underline{\theta}) - \log \pi(\alpha_n, k_n)$, and $p(\cdot)$ is a Gaussian probability distribution, and $\pi(\cdot)$ are mixture weighting coefficients, so that (up to a constant):

$$D(\alpha_n, k_n, \underline{\theta}, z_n) = -\log \pi(\alpha_n, k_n) + \frac{1}{2} \log \det \Sigma(\alpha_n, k_n) \qquad (9)$$
$$+ \frac{1}{2}[z_n - \mu(\alpha_n, k_n)]^\top \Sigma(\alpha_n, k_n)^{-1}[z_n - \mu(\alpha_n, k_n)].$$

Therefore, the parameters of the model are now

$$\underline{\theta} = \{\pi(\alpha, k), \mu(\alpha, k), \Sigma(\alpha, k), \ \alpha = 0, 1, \ k = 1 \ldots K\}, \qquad (10)$$

---

[1] Soft assignments of probabilities for each component to a given pixel might seem preferable as it would allow "Expectation Maximization" [Dempster et al. 1977] to be used; however that involves significant additional computational expense for what turns out to be negligible practical benefit.

i.e. the weights π, means μ and covariances Σ of the $2K$ Gaussian components for the background and foreground distributions. The smoothness term V is basically unchanged from the monochrome case (4) except that the contrast term is computed using Euclidean distance in colour space:

$$V(\underline{\alpha}, \mathbf{z}) = \gamma \sum_{(m,n) \in \mathbf{C}} [\alpha_n \neq \alpha_m] \exp -\beta \|z_m - z_n\|^2. \qquad (11)$$

3.2 Segmentation by iterative energy minimization

The new energy minimization scheme in GrabCut works iteratively, in place of the previous one-shot algorithm [Boykov and Jolly 2001]. This has the advantage of allowing automatic refinement of the opacities $\underline{\alpha}$, as newly labelled pixels from the T_U region of the initial trimap are used to refine the colour GMM parameters $\underline{\theta}$. The main elements of the GrabCut system are given in fig. 3. Step 1 is straightforward, done by simple enumeration of the k_n values for each pixel n. Step 2 is implemented as a set of Gaussian parameter estimation procedures, as follows. For a given GMM component k in, say, the foreground model, the subset of pixels $F(k) = \{z_n : k_n = k \text{ and } \alpha_n = 1\}$ is defined. The mean $\mu(\alpha, k)$ and covariance $\Sigma(\alpha, k)$ are estimated in standard fashion as the sample mean and covariance of pixel values in $F(k)$ and weights are $\pi(\alpha, k) = |F(k)| / \sum_k |F(k)|$, where $|\mathbf{S}|$ denotes the size of a set $\mathbf{S}$. Finally step 3 is a global optimization, using minimum cut, exactly as [Boykov and Jolly 2001].

The structure of the algorithm guarantees proper convergence properties. This is because each of steps 1 to 3 of *iterative minimisation* can be shown to be a minimisation of the total energy $\mathbf{E}$ with respect to the three sets of variables $\mathbf{k}$, $\underline{\theta}$, $\underline{\alpha}$ in turn. Hence $\mathbf{E}$ decreases monotonically, and this is illustrated in practice in fig. 4. Thus the algorithm is guaranteed to converge at least to a local minimum of $\mathbf{E}$. It is straightforward to detect when $\mathbf{E}$ ceases to decrease significantly, and to terminate iteration automatically.

Practical benefits of iterative minimisation. Figs. 2e and 2f illustrate how the additional power of iterative minimisation in GrabCut can reduce considerably the amount of user interaction needed to complete a segmentation task, relative to the one-shot graph cut [Boykov and Jolly 2001] approach. This is apparent in two ways. First the degree of user editing required, after initialisation and optimisation, is reduced. Second, the initial interaction can be simpler, for example by allowing incomplete labelling by the user, as described below.
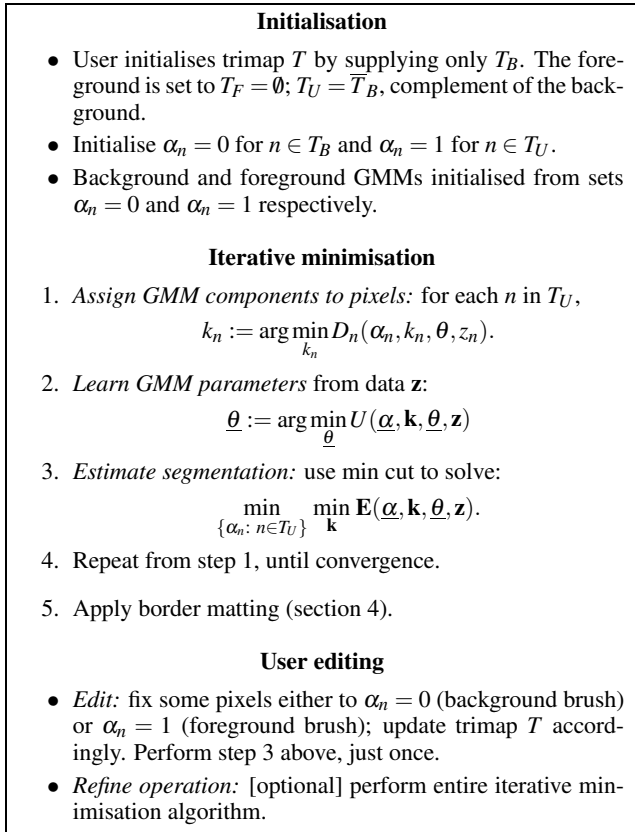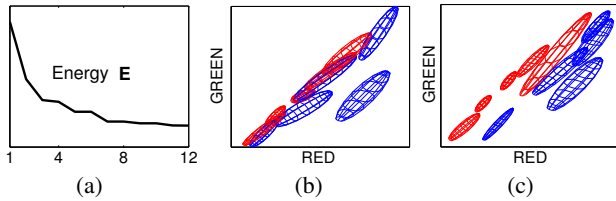
Figure 3: **Iterative image segmentation in GrabCut**

Figure 4: **Convergence of iterative minimization** for the data of fig. 2f. (a) The energy $\mathbf{E}$ for the llama example converges over 12 iterations. The GMM in RGB colour space (side-view showing R,G) at initialization (b) and after convergence (c). $K = 5$ mixture components were used for both background (red) and foreground (blue). Initially (b) both GMMs overlap considerably, but are better separated after convergence (c), as the foreground/background labelling has become accurate.

3.3 User Interaction and incomplete trimaps

Incomplete trimaps. The iterative minimisation algorithm allows increased versatility of user interaction. In particular, incomplete labelling becomes feasible where, in place of the full trimap T, the user needs only specify, say, the background region T_B, leaving $T_F = 0$. No hard foreground labelling is done at all. Iterative minimisation (fig. 3) deals with this incompleteness by allowing provisional labels on some pixels (in the foreground) which can subsequently be retracted; only the background labels T_B are taken to be firm — guaranteed not to be retracted later. (Of course a complementary scheme, with firm labels for the foreground only, is also a possibility.) In our implementation, the initial T_B is determined by the user as a strip of pixels around the outside of the marked rectangle (marked in red in fig. 2f).
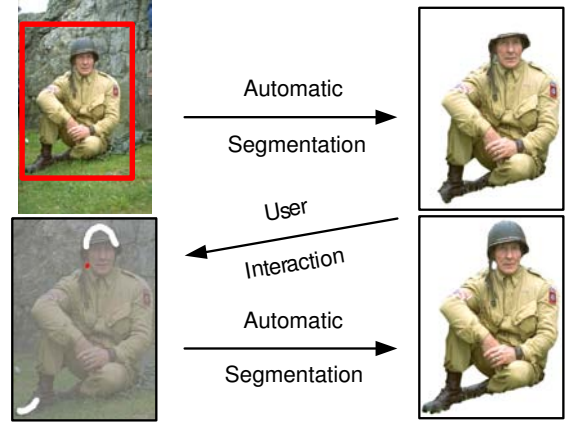

Figure 5: **User editing.** After the initial user interaction and segmentation (top row), further user edits (fig. 3) are necessary. Marking roughly with a foreground brush (white) and a background brush (red) is sufficient to obtain the desired result (bottom row).

Further user editing. The initial, incomplete user-labelling is often sufficient to allow the entire segmentation to be completed automatically, but by no means always. If not, further user editing is needed [Boykov and Jolly 2001], as shown in fig.5. It takes the form of brushing pixels, constraining them either to be firm foreground or firm background; then the minimisation step 3. in fig. 3 is applied. Note that it is sufficient to brush, roughly, just part of a wrongly labeled area. In addition, the optional "refine" operation of fig. 3 updates the colour models, following user edits. This propagates the effect of edit operations which is frequently beneficial. Note that for efficiency the optimal flow, computed by Graph Cut, can be re-used during user edits.

4 Transparency

Given that a matting tool should be able to produce continuous alpha values, we now describe a mechanism by which hard segmentation, as described above, can be augmented by "border matting", in which full transparency is allowed in a narrow strip around the hard segmentation boundary. This is sufficient to deal with the problem of matting in the presence of blur and mixed pixels along smooth object boundaries. The technical issues are: Estimating an alpha-map for the strip without generating artefacts, and recovering the foreground colour, free of colour bleeding from the background.

4.1 Border Matting

Border matting begins with a closed contour C, obtained by fitting a polyline to the segmentation boundary from the iterative hard segmentation of the previous section. A new trimap $\{T_B, T_U, T_F\}$ is computed, in which T_U is the set of pixels in a ribbon of width $\pm w$ pixels either side of C (we use $w = 6$). The goal is to compute the map α_n, $n \in T_U$, and in order to do this robustly, a strong model is assumed for the shape of the α-profile within T_U. The form of the model is based on [Mortensen and Barrett 1999] but with two important additions: regularisation to enhance the quality of the estimated α-map; and a dynamic programming (DP) algorithm for estimating α throughout T_U.

Let $t = 1, \ldots, T$ be a parameterization of contour C, periodic with period T, as curve C is closed. An index $t(n)$ is assigned to each pixel $n \in T_U$, as in fig. 6(c). The α-profile is taken to be a soft step-function g (fig. 6c): $\alpha_n = g\left(r_n; \Delta_{t(n)}, \sigma_{t(n)}\right)$, where r_n is a signed distance from pixel n to contour C. Parameters Δ, σ determine the centre and width respectively of the transition from 0 to 1 in the
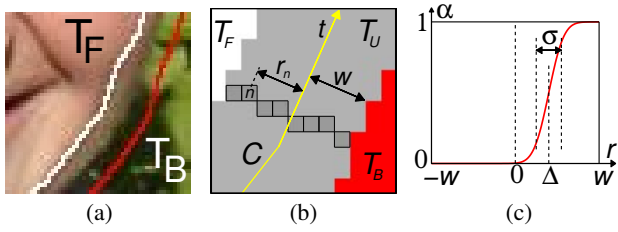
Figure 6: **Border matting.** (a) Original image with trimap over-laid. (b) Notation for contour parameterisation and distance map. Contour C (yellow) is obtained from hard segmentation. Each pixel in T_U is assigned values (integer) of contour parameter t and distance r_n from C. Pixels shown share the same value of t. (c) Soft step-function for α-profile g, with centre Δ and width σ.

α-profile. It is assumed that all pixels with the same index t share values of the parameters Δ_t, σ_t.

Parameter values $\Delta_1, \sigma_1, \ldots, \Delta_T, \sigma_T$ are estimated by minimizing the following energy function using DP over t:

$$E = \sum_{n \in T_U} \tilde{D}_n(\alpha_n) + \sum_{t=1}^{T} \tilde{V}(\Delta_t, \sigma_t, \Delta_{t+1}, \sigma_{t+1}) \qquad (12)$$

where $\tilde{V}$ is a smoothing regularizer:

$$\tilde{V}(\Delta, \sigma, \Delta', \sigma') = \lambda_1 (\Delta - \Delta')^2 + \lambda_2 (\sigma - \sigma')^2, \qquad (13)$$

whose role is to encourage α-values to vary smoothly as t increases, along the curve C (and we take $\lambda_1 = 50$ and $\lambda_2 = 10^3$). For the DP computation, values of Δ_t are discretised into 30 levels and σ_t into 10 levels. A general smoothness term $\tilde{V}$ would require a quadratic time in the number of profiles to move from t to $t+1$, however our regularizer allows a linear time algorithm using distance transforms [Rucklidge 1996]. If the contour C is closed, minimization cannot be done exactly using single-pass DP and we approximate by using two complete passes of DP, assuming that the first pass gives the optimal profile for $t = T/2$.

The data term is defined as

$$\tilde{D}_n(\alpha_n) = -\log \mathbf{N}\left(z_n; \mu_{t(n)}(\alpha_n), \Sigma_{t(n)}(\alpha_n)\right) \qquad (14)$$

where $\mathbf{N}(z; \mu, \Sigma)$ denotes a Gaussian probability density for z with mean μ and covariance Σ. Mean and covariance for (14) are defined for matting as in [Ruzon and Tomasi 2000]:

$$\mu_t(\alpha) = (1-\alpha)\mu_t(0) + \alpha\mu_t(1) \qquad (15)$$
$$\Sigma_t(\alpha) = (1-\alpha)^2 \Sigma_t(0) + \alpha^2 \Sigma_t(1).$$

The Gaussian parameters $\mu_t(\alpha)$, $\Sigma_t(\alpha)$, $\alpha = 0, 1$ for foreground and background are estimated as the sample mean and covariance from each of the regions $\mathbf{F}_t$ and $\mathbf{B}_t$ defined as $\mathbf{F}_t = S_t \cap T_F$ and $\mathbf{B}_t = S_t \cap T_B$, where S_t is a square region of size $L \times L$ pixels centred on the segmentation boundary C at t (and we take $L = 41$).

4.2 Foreground estimation

The aim here is to estimate foreground pixel colours without colours bleeding in from the background of the source image. Such bleeding can occur with Bayes matting because of the probabilistic algorithm used which aims to strip the background component from mixed pixels but cannot do so precisely. The residue of the stripping process can show up as colour bleeding. Here we avoid this by stealing pixels from the foreground T_F itself. First the Bayes matte algorithm [Chuang et al. 2001, eq. (9)] is applied to obtain an

estimate of foreground colour $\hat{f}_n$ on a pixel $n \in T_U$. Then, from the neighbourhood $\mathbf{F}_{t(n)}$ as defined above, the pixel colour that is most similar to $\hat{f}_n$ is stolen to form the foreground colour f_n. Finally, the combined results of border matting, using both regularised alpha computation and foreground pixel stealing, are illustrated in fig. 7.

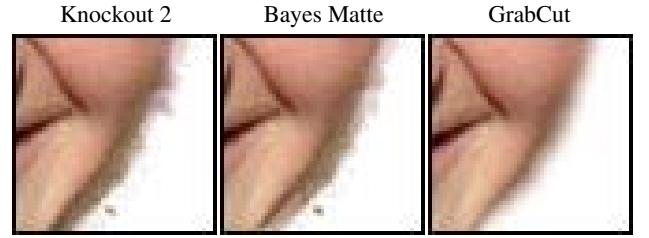

Knockout 2   Bayes Matte   GrabCut

Figure 7: **Comparing methods for border matting.** Given the border trimap shown in fig. 6, GrabCut obtains a cleaner matte than either Knockout 2 or Bayes matte, due to the regularised α-profile.

5 Results and Conclusions

Various results are shown in figs. 1, 8. The images in fig. 1 each show cases in which the bounding rectangle alone is a sufficient user interaction to enable foreground extraction to be completed automatically by GrabCut. Fig. 8 shows examples which are increasingly more difficult. Failures, in terms of significant amount of user interactions, can occur in three cases: (i) regions of low contrast at the transition from foreground to background (e.g. black part of the butterfly's wing fig. 8); (ii) camouflage, in which the true foreground and background distributions overlap partially in colour space (e.g. soldier's helmet fig. 5); (iii) background material inside the user rectangle happens not to be adequately represented in the background region (e.g. swimmer fig. 8). User interactions for the third case can be reduced by replacing the rectangle with a lasso[2]. The fox image (fig. 8) demonstrates that our border matting method can cope with moderately difficult alpha mattes. For difficult alpha mattes, like the dog image (fig. 8), the matting brush is needed.

The operating time is acceptable for an interactive user interface, e.g. a target rectangle of size 450x300 pixels (butterfly fig. 8) requires $0.9sec$ for initial segmentation and $0.12sec$ after each brush stroke on a 2.5 GHz CPU with 512 MB RAM.

Comparison of Graph Cut [Boykov et. al. 2001] and GrabCut. In a first experiment the amount of user interaction for 20 segmentation tasks were evaluated. In 15 moderately difficult examples (e.g. flowers fig. 1) GrabCut needed significantly fewer interactions. Otherwise both methods performed comparably[2].

In a second experiment we compare GrabCut using a single "outside" lasso (e.g. red line in fig. 2c (top)) with Graph Cut using two lassos, outer and inner (e.g. red and white lines in fig. 2c (top)). Even with missing foreground training data, GrabCut performs at almost the same level of accuracy as Graph Cut. Based on a ground truth database of 50 images[2], the error rates are $1.36 \pm 0.18\%$ for Graph Cut compared with $2.13 \pm 0.19\%$ for GrabCut.

In conclusion, a new algorithm for foreground extraction has been proposed and demonstrated, which obtains foreground alpha mattes of good quality for moderately difficult images with a rather modest degree of user effort. The system combines hard segmentation by iterative graph-cut optimisation with border matting to deal with blur and mixed pixels on object boundaries.

Image credits. Images in fig. 5, 8 and flower image in fig. 1 are from Corel Professional Photos, copyright 2003 Microsoft Research and its licensors, all rights reserved. The llama image (fig. 1, 2) is courtesy of Matthew Brown.

—————————
[2]see www.research.microsoft.com/vision/cambridge/segmentation/
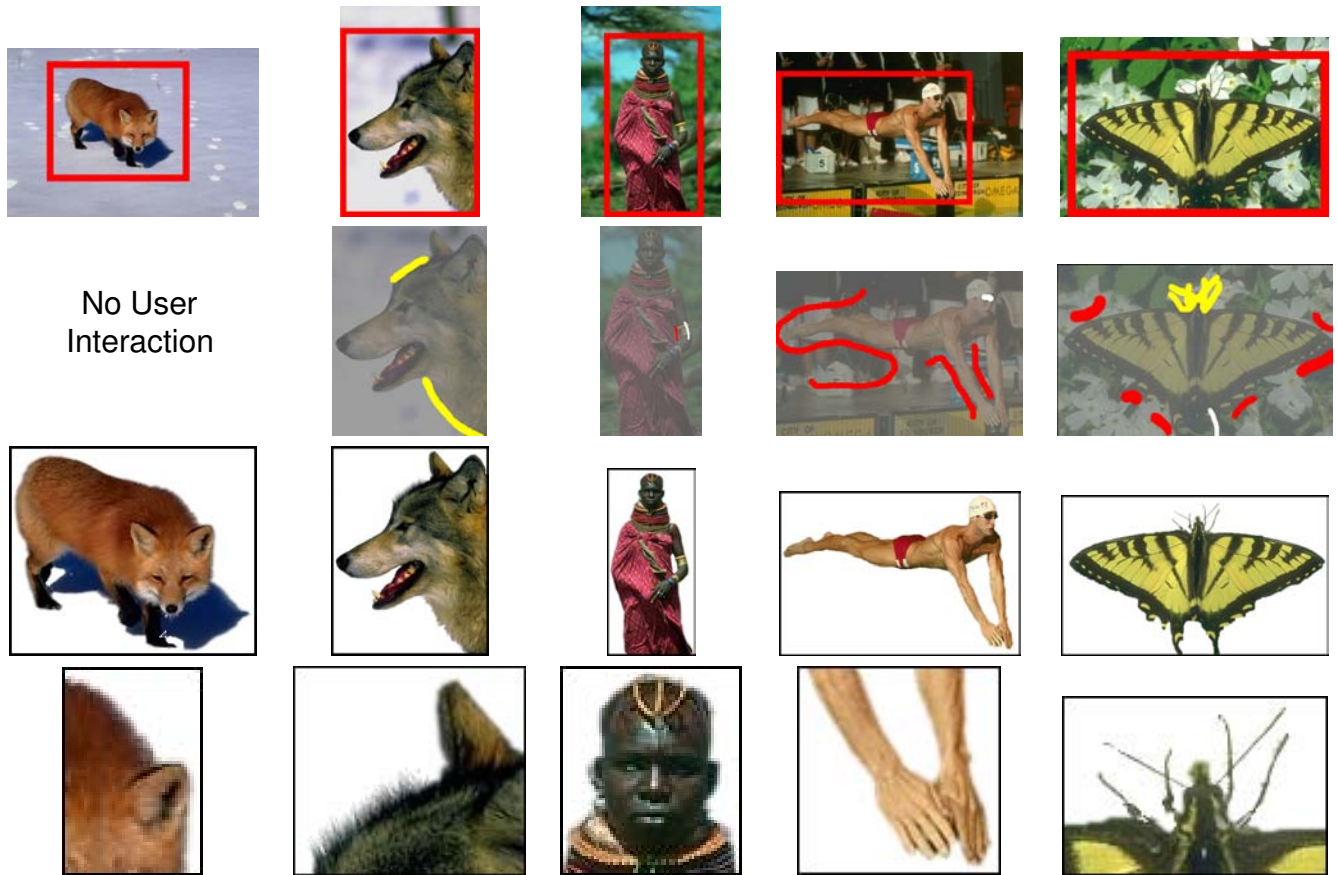
No User
Interaction

Figure 8: **Results using GrabCut.** The first row shows the original images with superimposed user input (red rectangle). The second row displays all user interactions: red (background brush), white (foreground brush) and yellow (matting brush). The degree of user interaction increases from left to right. The results obtained by GrabCut are visualized in the third row. The last row shows zoomed portions of the respective result which documents that the recovered alpha mattes are smooth and free of background bleeding.

References

ADOBE SYSTEMS INCORP. 2002. Adobe Photoshop User Guide.

BLAKE, A., ROTHER, C., BROWN, M., PEREZ, P., AND TORR, P. 2004. Interactive Image Segmentation using an adaptive GMMRF model. In *Proc. European Conf. Computer Vision*.

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

BOYKOV, Y., AND KOLMOGOROV, V. 2003. Computing Geodesics and Minimal Surfaces via Graph Cut. In *Proc. IEEE Int. Conf. on Computer Vision*.

CASELLES, V., KIMMEL, R., AND SAPIRO, G. 1995. Geodesic active contours. In *Proc. IEEE Int. Conf. on Computer Vision*.

CHUANG, Y.-Y., CURLESS, B., SALESIN, D., AND SZELISKI, R. 2001. A Bayesian approach to digital matting. In *Proc. IEEE Conf. Computer Vision and Pattern Recog.*, CD–ROM.

COREL CORPORATION. 2002. Knockout user guide.

DEMPSTER, A., LAIRD, M., AND RUBIN, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc. B. 39*, 1–38.

GREIG, D., PORTEOUS, B., AND SEHEULT, A. 1989. Exact MAP estimation for binary images. *J. Roy. Stat. Soc. B. 51*, 271–279.

KASS, M., WITKIN, A., AND TERZOPOULOS, D. 1987. Snakes: Active contour models. In *Proc. IEEE Int. Conf. on Computer Vision*, 259–268.

KOLMOGOROV, V., AND ZABIH, R. 2002. What energy functions can be minimized via graph cuts? In *Proc. ECCV*. CD–ROM.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut Textures: Image and Video Synthesis Using Graph Cuts. *Proc. ACM Siggraph*, 277–286.

MORTENSEN, E., AND BARRETT, W. 1995. Intelligent scissors for image composition. *Proc. ACM Siggraph*, 191–198.

MORTENSEN, E., AND BARRETT, W. 1999. Tobogan-based intelligent scissors with a four parameter edge model. In *Proc. IEEE Conf. Computer Vision and Pattern Recog.*, vol. 2, 452–458.

RUCKLIDGE, W. J. 1996. *Efficient visual recognition using the Hausdorff distance*. LNCS. Springer-Verlag, NY.

RUZON, M., AND TOMASI, C. 2000. Alpha estimation in natural images. In *Proc. IEEE Conf. Comp. Vision and Pattern Recog.*