Steerable Semi-automatic Segmentation of Textured Images[⋆]

Branislav Mičušík and Allan Hanbury

Pattern Recognition and Image Processing Group,
Institute of Computer Aided Automation,
Vienna University of Technology,
Favoritenstraße 9/1832, A-1040 Vienna, Austria
{micusik, hanbury}@prip.tuwien.ac.at

Abstract. This paper generalizes the interactive method for region segmentation of grayscale images based on graph cuts by Boykov & Jolly (ICCV 2001) to colour and textured images. The main contribution lies in incorporating new functions handling colour and texture information into the graph representing an image, since the previous method works for grayscale images only. The suggested method is semi-automatic since the user provides additional constraints, i.e. s(he) establishes some seeds for foreground and background pixels. The method is steerable by a user since the change in the segmentation due to adding or removing seeds requires little computational effort and hence the evolution of the segmentation can easily be controlled by the user. The foreground and background regions may consist of several isolated parts. The results are presented on some images from the Berkeley database.

1 Introduction

Fully automatic image segmentation is still an open problem in computer vision. An ideal algorithm would take a single image as an input and give the image segmented into semantically meaningful, non-overlapping regions as the output. However, single image segmentation is ill-posed problem and the usual result is either over- or under-segmentation. Moreover, measuring the goodness of segmentations in general is an unsolved problem. Obtaining absolute ground truth is difficult since different people produce different manual segmentations of the same scene [8].

There are many papers dealing with automatic segmentation. We mention only the state-of-the-art work based on normalized cuts [13] for segmenting the image into many non-overlapping regions. This method uses graph cuts as we do, but a modification is introduced, i.e. normalized graph cuts together with an approximate closed-form solution. However, the boundaries of detected regions often do not follow the true boundaries of the objects. The work [14] is a follow-up to [13] where the segmentation is done at various scales.

⋆ This work was supported by the Austrian Science Foundation (FWF) under grant SESAME (P17189-N04), and the European Union Network of Excellence MUSCLE (FP6-507752).

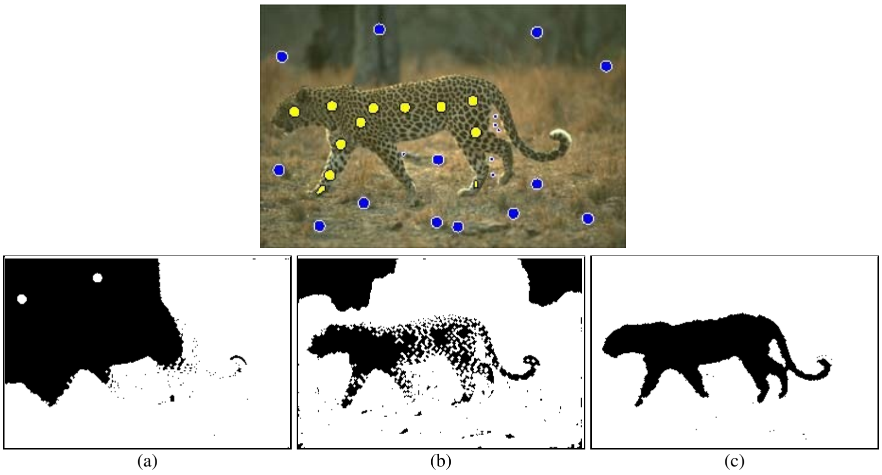


Fig. 1. Image segmentation of the leopard image at the top with user-specified foreground (bright/yellow) and background seeds (dark/blue). For the segmentation (a) grayscale [3], (b) colour, and (c) colour+texture information were used respectively

One possibility to partially avoid the ill-posed problem of image segmentation is to use additional constraints. Such constraints can be *i)* motion in the image caused either by camera motion or by motion of objects in the scene [12, 15, 1], or *ii)* specifying the foreground object properties [3, 11, 2].

The motion assumption is used in video matting [12, 15, 1]. The main focus is finding the opacity/transparency of boundary pixels of the foreground object. In our case we perform binary segmentation, i.e. the pixel can belong either to the foreground or the background. No fuzzy memberships are allowed.

In this paper we focus on single image segmentation. We follow the idea given in [3] of interactive segmentation where the user has to specify some pixels belonging to the foreground and to the background. Such labeled pixels give a strong constraint for further segmentation based on min-cut/max-flow algorithm given in [4]. However, the method [3] was designed for grayscale images and thus most of the information is thrown away. In Fig. 1a, one sees the poor result obtained for a textured image segmented using only the grayscale information. By reducing the colour image to a grayscale one, the different colour regions can transform to the same grayscale intensity. Fig. 1b shows how adding colour information helps to achieve a better result. However, many regions contain texture (most natural objects). Taking texture into account helps to improve the final segmentation even more, see Fig. 1c.

The paper [2] uses the segmentation technique [3] as we do. They suggest a method for learning parameters of colour and contrast models left for the user in the method in [3]. They use the Gaussian mixture Markov random field framework. However, contrary to this paper, they do not handle texture information.

In [16] the spatial coherence of the pixels together with standard local measurements (intensity, colour) is handled. They propose an energy function that operates simultaneously in feature space and in image space. Some forms of such an energy function are studied in [6]. In our work we follow a similar strategy. However, we define the neighborhood relation through brightness, colour and texture gradients introduced in [7, 9].

In [11] the boundary of a textured foreground object is found by minimization (through the evolution of the region contour) of energies inside and outside the region in the context of the Geodetic Active Region framework. However, the texture information for the foreground has to be specified by the user. In [10] the user interaction is omitted by finding representative colours by fitting a mixture of Gaussian elements to the image histogram. However, such technique cannot be used for textured images.

The main contribution of this paper lies in incorporating brightness, colour and texture cues based on the work [7, 9] into the segmentation method [3] based on the maximal flow algorithm. Second, we introduce a new penalty function derived through the Bayesian rule to measure likelihood of a pixel being foreground or background. The proposed method allows one to segment textured images controlled by user interaction.

The structure of the paper is as follows. In Sec. 2, brightness, colour and texture gradients are briefly described. In Sec. 3, a segmentation based on the graph cut algorithm is outlined together with the new energy functions. Finally, the results and summary conclude the work.

2 Boundary Detection

Boundary detection is a difficult task, as it should work for a wide range of images, i.e. for images of human-made environments and for natural images. Our main emphasis is put on boundaries at the changes of different textured regions and not local changes inside one texture. This is complicated since there are usually large responses of edge detectors inside the texture. To detect boundaries in images correctly, the colour changes and texturedness of the regions have to be taken into account.

In this work we use as a cue the brightness, colour, and texture gradients introduced in [7, 9]. We shortly outline the basic paradigm.

2.1 Brightness and Colour Gradient

First, the RGB space is converted into the CIELAB $L^*a^*b^*$ space. Each of the three channels is treated separately and finally merged together with the texture gradient (will be explained).

Second, at each pixel location (x, y) in the image, a circle of radius r is created, and divided along the diameter at orientation θ. The gradient function $G(x, y, \theta, r)$ compares the contents (histograms) of the two resulting disc halves. A large difference between the disc halves indicates a discontinuity in the image along the disc diameter. In our experiments we used 8 orientations, every $45°$, the radius for the L channel $r_L = d/100$, for the a channel $r_a = d/200$ and for the b channel $r_b = d/200$. d is the length of the diagonal of the image in pixels. We adopt the values used in [9]. The half-disc regions are described by histograms g_i, h_i which are compared using the χ^2 histogram difference operator

$$\chi^2(g, h) = \frac{1}{2} \sum_{i}^{N_b} \frac{(g_i - h_i)^2}{g_i + h_i}, \tag{1}$$

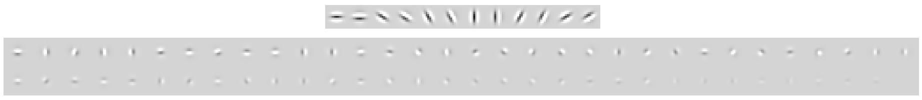


Fig. 2. Top: Filter bank for one scale. Bottom: Universal textons sorted by their norms

where N_b is the number of bins in the histograms, here for brightness and colour gradient $N_b = 32$. For one pixel there are as many numbers as orientations of θ (in our case 8). The gradient at each pixel is the maximal number chosen over all orientations. To obtain more robust results suppression of the non-maxima and the use of parabolic interpolation is advisable. The reader is referred to [9] for more details.

After this step a gradient for every channel, i.e. $G^L(x, y)$, $G^a(x, y)$, $G^b(x, y)$ is obtained.

2.2 Texture Gradient

By the texture gradient we mean the gradient computed on the image in the texton domain to capture the variation in intensities in some local neighborhood. The gradient is not related to surface orientation as sometimes used in literature.

To evaluate the texture gradient, we make use of the oriented filter bank, depicted at the top of Fig. 2. The filters are based on rotated copies of a Gaussian derivative and its Hilbert transform. More precisely, even- and odd-symmetric filters, respectively, are written as follows

$$\begin{aligned} f_1(x, y) &= N''_{0,\sigma_1}(y) N_{0,\sigma_2}(x), \\ f_2(x, y) &= Hilbert(f_1(x, y)), \end{aligned} \tag{2}$$

where N_{0,σ_1} is Gaussian with zero mean and variance σ_1. N''_{0,σ_1} stands for the second derivative. The ratio $\sigma_2 : \sigma_1$ is a measure of the elongation of the filter. We used the ratio $\frac{\sigma_2}{\sigma_1} = 2$. The image is convolved with such a bank of linear filters. After that each pixel contains a feature vector with responses to all filters in the filter bank. In our case, we used 24 filters (odd and even symmetric filters with 6 orientations and 2 scales, see the top of Fig. 2). Center-surround filters as in [7] could be added to the filterbank.

The pixels are then clustered, e.g. by K-means, in filter response feature space. The dominant K clusters are called *textons*. Alternatively, as we did, the universal textons (obtained from many representative images in Berkeley's database) can be used, see the bottom image of Fig. 2. Then each pixel is assigned to the closest "universal" texton. By this step the image range, usually $0 - 255$, is transformed to the range $1 - K$, where K is the number of textons (in our case 64).

The same strategy based on half-discs with 6 orientations and comparing the histograms, as was explained for the brightness and colour gradients in the previous subsection, is applied to the texton image. The number of histogram bins in Eq. (1) is now the number of textons.

After this step a texture gradient $G^T(x, y)$ is obtained.

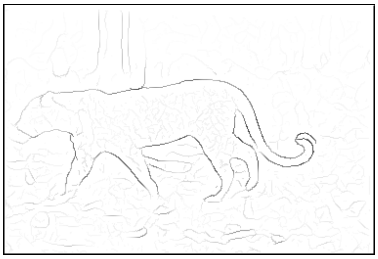


Fig. 3. Combined boundary probability using colour + texture gradient of the leopard image. Black points stand for high, white for low boundary probability

2.3 Combined Boundary Probability

The final step for the boundary detection in textured images is to merge the above gradients to obtain a single value for each pixel.

We begin with a vector composed of the brightness, colour, and texture gradients,

$$\mathbf{x}(x,y) = [1,\ G^L(x,y),\ G^a(x,y),\ G^b(x,y),\ G^T(x,y)]^\top . \tag{3}$$

To define the final probability for a pixel at position (x, y) to be a boundary, a sigmoid is used [7]

$$p_b(x,y) = \frac{1}{1 + e^{-\mathbf{x}^\top \mathbf{b}}} , \tag{4}$$

where the constant vector $\mathbf{b}$ consists of weights for each partial gradient. If there is no boundary change in a pixel at position (x, y), the vector $\mathbf{x} = (1,\ 0,\ 0,\ 0,\ 0)^\top$ and $\mathbf{x}^\top \mathbf{b} = b_1$. The "1" is at the beginning of the vector $\mathbf{x}$ hence allows one to control the weight in the "no boundary" case through the b_1 in the vector $\mathbf{b}$.

The method for obtaining the weights in $\mathbf{b}$, i.e. combining the information from all gradients in an optimal way, is suggested in [9]. They used human labeled images from the Berkeley database as ground truth [8]. In our implementation we used the $\mathbf{b}$ provided with the source code on the web by the authors [9].

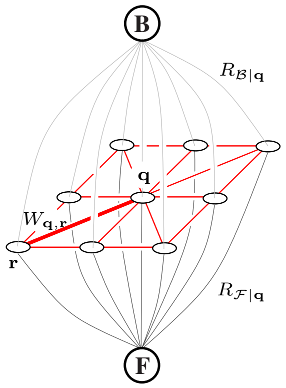
The "0" value for p_b in Eq. (4) indicates no boundary, the "1" value indicates a boundary, i.e. a change of colour or texture, in the image with maximal confidence. See Fig. 3 for the combined boundary probability of the image in Fig. 1.

3 Segmentation

We used a segmentation technique based on the interactive graph cuts method first introduced in [3]. There exists a very efficient algorithm for finding min-cut/max-flow in a graph [4]. We introduced new penalties on edges in the graph based on a RGB colour cue and on a combined boundary cue, respectively.

3.1 Building the Graph

The general framework for building the graph is depicted in Fig. 4. The graph is shown for a 9 pixel image and an 8-point neighborhood $\mathcal{N}$. For general images, the graph has


B
$R_{\mathcal{B}|\mathbf{q}}$
q
$W_{\mathbf{q},\mathbf{r}}$
r
$R_{\mathcal{F}|\mathbf{q}}$
F

edge	cost	region
$\{\mathbf{q},\mathbf{r}\}$	$W_{\mathbf{q},\mathbf{r}}$	$\{\mathbf{q},\mathbf{r}\} \in \mathcal{N}$
$\{\mathbf{q},F\}$	$\lambda\, R_{\mathcal{F}\|\mathbf{q}}$	$\mathbf{q} \notin \mathcal{F} \cup \mathcal{B}$
	K	$\mathbf{q} \in \mathcal{F}$
	0	$\mathbf{q} \in \mathcal{B}$
$\{\mathbf{q},B\}$	$\lambda\, R_{\mathcal{B}\|\mathbf{q}}$	$\mathbf{q} \notin \mathcal{F} \cup \mathcal{B}$
	0	$\mathbf{q} \in \mathcal{F}$
	K	$\mathbf{q} \in \mathcal{B}$

Fig. 4. Left: Graph representation for 9 pixel image. Right: Table defining the costs of graph edges. K and λ are constants described in the text

as many nodes as pixels plus two extra nodes labeled F, B, and the neighborhood is larger.

Each node in the graph is connected to the two extra nodes F, B. It allows the incorporation of the information provided by the user and sets a penalty for each pixel being foreground or background. The user specifies two disjoint sets $\mathcal{F}$ and $\mathcal{B}$ containing samples of foreground and background pixels. If, for instance, the image point $\mathbf{q}$ is marked as belonging to the foreground then there is a maximum weight K on the edge $\{\mathbf{q}, F\}$ and zero weight on the edge $\{\mathbf{q}, B\}$. K is some large number larger than $K_{min} = 1 + \max_{\mathbf{q}} \sum_{\mathbf{r}:\{\mathbf{q},\mathbf{r}\}\in\mathcal{N}} W_{\mathbf{q},\mathbf{r}}$.

The regional penalty of a point $\mathbf{q}$ not marked by the user as being foreground $\mathcal{F}$ or background $\mathcal{B}$ is defined as follows

$$\begin{aligned} R_{\mathcal{F}|\mathbf{q}} &= -\ln p(\mathcal{B}|\mathbf{c_q}) \\ R_{\mathcal{B}|\mathbf{q}} &= -\ln p(\mathcal{F}|\mathbf{c_q}), \end{aligned} \tag{5}$$

where $\mathbf{c_q} = (c_r,\ c_g,\ c_b)^\top$ stands for a vector in $\mathbb{R}^3$ of RGB values at the pixel $\mathbf{q}$. To compute the posterior probabilities in Eq. (5) we used the Bayesian rule as follows

$$p(\mathcal{B}|\mathbf{c_q}) = \frac{p(\mathbf{c_q}|\mathcal{B})\, p(\mathcal{B})}{p(\mathbf{c_q})} = \frac{p(\mathbf{c_q}|\mathcal{B})\, p(\mathcal{B})}{p(\mathcal{B})\, p(\mathbf{c_q}|\mathcal{B}) + p(\mathcal{F})\, p(\mathbf{c_q}|\mathcal{F})}. \tag{6}$$

We demonstrate it on $p(\mathcal{B}|\mathbf{c_q})$, for $p(\mathcal{F}|\mathbf{c_q})$ it is analogical.

We do not know a priori the probabilities $p(\mathcal{F})$ and $p(\mathcal{B})$ of the foreground and background regions, i.e. how large the foreground region is compared to the background one. Thus, we fixed them to $p(\mathcal{F}) = p(\mathcal{B}) = 0.5$ and Eq. (6) then reduces to

$$p(\mathcal{B}|\mathbf{c_q}) = \frac{p(\mathbf{c_q}|\mathcal{B})}{p(\mathbf{c_q}|\mathcal{B}) + p(\mathbf{c_q}|\mathcal{F})}, \tag{7}$$

where the prior probabilities are

$$p(\mathbf{c_q}|\mathcal{F}) = f^r_{c_r} \cdot f^g_{c_g} \cdot f^b_{c_b}, \quad \text{and} \quad p(\mathbf{c_q}|\mathcal{B}) = b^r_{c_r} \cdot b^g_{c_g} \cdot b^b_{c_b},$$

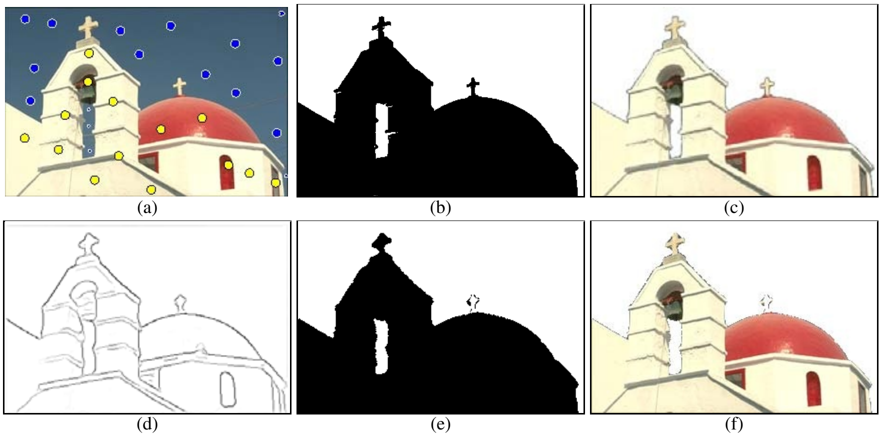


Fig. 5. Church image. (a) Input image with specified foreground/background seeds. (b,c) Segmentation using colour cue. (d) Combined boundary probability. (e,f) Segmentation using colour and texture gradient

where $f_i^{\{r,g,b\}}$, resp. $b_i^{\{r,g,b\}}$, represents the foreground, resp. the background histogram of each colour channel separately at the ith bin learned from seed pixels. Here the histograms are represented in RGB colour space, nevertheless another colour space, e.g. L*a*b*, can be used. We used 64 bins for each colour channel. For better performance we smoothed the histograms using a one-dimensional Gaussian kernel.

In an implementation one should take into account the possibility of a zero value of $p(\mathcal{B}|\mathbf{c_q})$ in Eq. (5) and thus avoid an overflow. In such a case $R_{\mathcal{F}|\mathbf{q}} = K$.

The edge weights of neighborhood $\mathcal{N}$ are encoded in the matrix $W_{\mathbf{q},\mathbf{r}}$, which is not necessarily symmetric. Setting the values of these weights is discussed in the following subsections. The size and density of the neighborhood are controlled through two parameters. We used a neighborhood window of size 21×21 with sample rate 0.3, i.e. only a randomly selected 30% of all pixels in the window are used. Using only a fraction of pixels in the window reduces the computational demand and thus allows the use of larger windows while preserving the spatial relations.

3.2 Segmentation Using RGB Colour Cue

The simplest straightforward modification of the weight function in [3] is the augmentation of the penalty function to take colour into account. The weight matrix is chosen as follows

$$W_{\mathbf{q},\mathbf{r}} = e^{-\frac{\|\mathbf{c_q}-\mathbf{c_r}\|^2}{\sigma_1}} \cdot \frac{1}{\|\mathbf{q}-\mathbf{r}\|}, \tag{8}$$

where $\mathbf{c_q}$ is the RGB vector of a point at the position $\mathbf{q}$ (as in Eq. (5)). σ_1 is a parameter (we used $\sigma_1 = 0.02$ in all our experiments).

The penalty in Eq. (8) is good only for textureless images, as in Fig. 5. The next section suggests a more general approach using colour and texture.

3.3 Segmentation Using Combined Boundary Cue

A more general approach to define graph weights is to incorporate the combined boundary probability from Sec. 2.3. The neighborhood penalty of two pixels is defined as

$$W_{\mathbf{q},\mathbf{r}} = \left(e^{-\frac{g(\mathbf{q},\mathbf{r})^2}{\sigma_2}} \right)^2 , \tag{9}$$

where σ_2 is a parameter (we used $\sigma_2 = 0.08$ in all our experiments) and

$$g(\mathbf{q},\mathbf{r}) = p_b(\mathbf{q}) + \max_{\mathbf{s} \in \mathcal{L}_{\mathbf{q},\mathbf{r}}} p_b(\mathbf{s}) , \tag{10}$$

where $p_b(\mathbf{q})$ is the combined boundary probability described in Sec. 2.3 and $\mathcal{L}_{\mathbf{q},\mathbf{r}} = \{\mathbf{x} \in \mathbb{R}^2 \colon \mathbf{x} = \mathbf{q} + k(\mathbf{r} - \mathbf{q}), k \in \langle 0, 1 \rangle\}$ is a set of points on a line from the point $\mathbf{q}$ (exclusive) to the point $\mathbf{r}$ (inclusive). We used the DDA line algorithm to discretize the line. The penalty in Eq. (10) follows the idea that there is a large weight if the line connecting two points crosses an edge in the combined boundary probability image. The value of the weight corresponds to the strength of the edge. If there is no edge between the points the weight approaches zero.

4 Experiments

The segmentation method was implemented in MATLAB. Some of the most time consuming operations (creating the graph edge weights) were implemented in C and interfaced with MATLAB through mex-files. We used with advantage the sparse matrices directly offered by MATLAB. We used the online available C++ implementations of the min-cut algorithm [4] and some MATLAB code for colour and texture gradient computation [7]. The parameters like number of histogram bins (64), neighborhood window size (21x21), sample rate (0.3), and sensitivities ($\sigma_1 = 0.02$, $\sigma_2 = 0.08$) were obtained experimentally for giving the best performance on a large database of images.

The most time consuming part of the segmentation process is creating the weight matrix W. However, the matrix W is created only once and adding some new seeds (user interaction) changes only the penalties $R_{\mathcal{F}|\mathbf{q}}$, $R_{\mathcal{B}|\mathbf{q}}$ which requires a minimum amount of time. Hence the segmentation method can be designed as an interactive method, i.e. the user can interactively add new seed points and thus control the final segmentation. Once the graph is built, finding the min-cut takes 2 – 5 seconds on a 250×375 image running on a Pentium 4@2.8 GHz.

The first experiment shows the performance of both suggested weights, Eq. (8) and Eq. (9). The church image in Fig. 5 is a colour image with no significant texture. In this case, using the same seed points, the method based on the RGB colour space gives a comparable result to the method based on the colour + texture gradient. Notice the missing cross in the second method. The user can of course add extra seed points on the cross and the problem would be solved. However, we wanted to show that the penalty based on colour can sometimes give better results than using color + texture gradient and is therefore useful in certain applications, especially if execution time is the main criterion. Segmentation by the first method took 13 seconds, by the second method 97

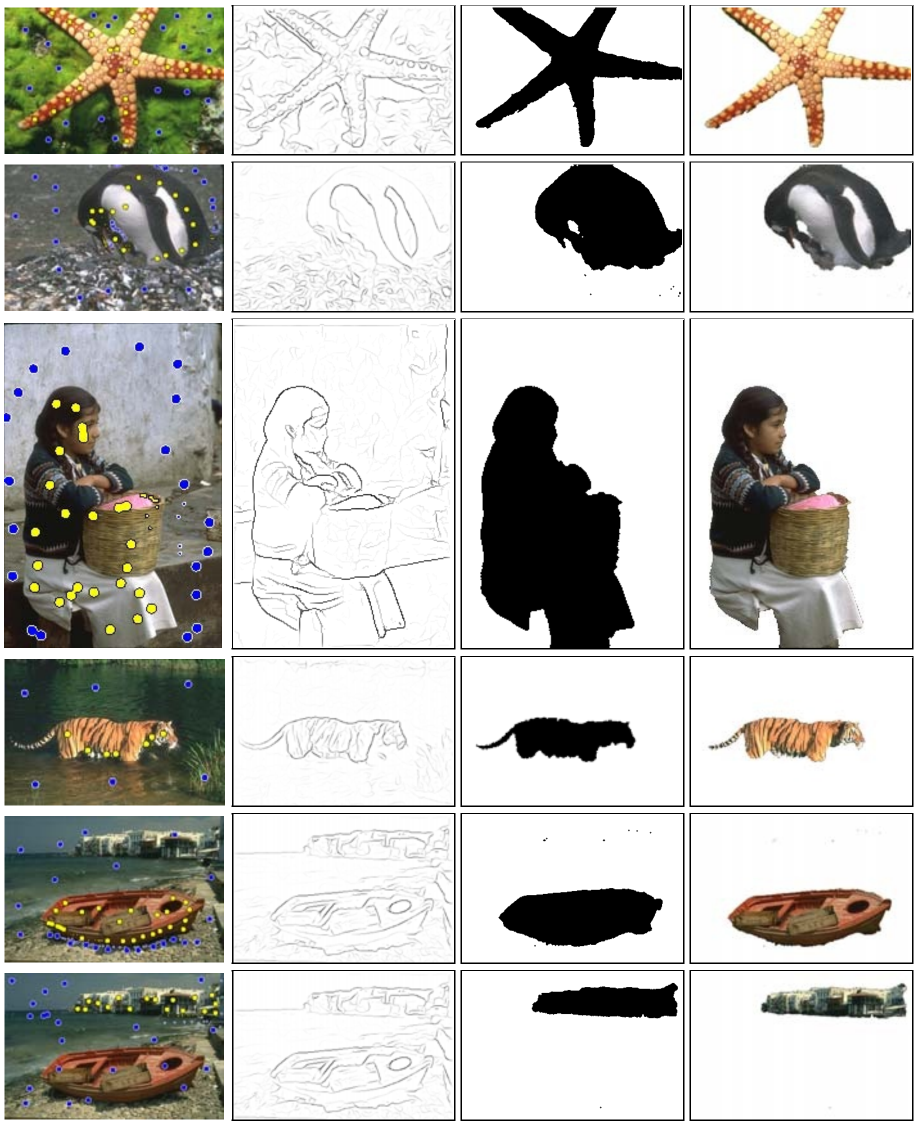


Fig. 6. Results. 1st column: original image with foreground (bright/yellow) and background (dark/blue) seeds. 2nd column: combined boundary probability. 3rd column: binary segmentation. 4th column: segmentation with masked original image

seconds. However, the implementation of the texture gradient in C would dramatically speed up the computation time of the second approach.

For textured images, as in Fig. 1, the method taking into account texture information gives the best result. Other results are shown in Fig. 6 on various images from the Berkeley database [5]. On the last "boat" image, it is shown that it depends on the user to specify what the object of interest in the image will be. It enables one to segment the image into many regions. In our case we first segmented the boat, then the houses at the back.

5 Conclusion

We improved the method [3] based on graph cuts by incorporating the brightness, colour, and texture gradient based on [7] into the graph edge weights. We introduced a new penalty function derived through the Bayesian rule to measure the pixel likelihood of being foreground or background. The proposed method is semi-automatic and provides segmentation into foreground and background objects (which may consist of several isolated parts). The method is interactive since adding or removing seeds takes little computational effort and hence the evolution of the segmentation can easily be controlled by the user.

References

1. N. Apostoloff and A. Fitzgibbon. Bayesian estimation of layers from multiple images. In *Proc. CVPR*, volume 1, pages 407–414, 2004.
2. A. Blake, C. Rother, M. Brown, P. P+rez, and P. H. S. Torr. Interactive image segmentation using an adaptive GMMRF model. In *Proc. ECCV*, volume 1, pages 428–441, 2004.
3. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *Proc. ICCV*, pages 105–112, 2001.
4. Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 26(9):1124–1137, 2004.
5. `http://www.cs.berkeley.edu/projects/vision/grouping/segbench/`.
6. V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *PAMI*, 26(2):147–159, 2004.
7. J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *IJCV*, 43(1):7–27, 2001.
8. D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. ICCV*, pages 416–425, 2001.
9. D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549, 2004.
10. N. Paragios and R. Deriche. Coupled geodesic active regions for image segmentation: A level set approach. In *Proc. ECCV*, volume II, pages 224–240, 2000.
11. N. Paragios and R. Deriche. Geodesic active regions and level set methods for supervised texture segmentation. *IJCV*, 46(3):223–247, 2002.
12. M. Ruzon and C. Tomasi. Alpha estimation in natural images. In *Proc. CVPR*, volume 1, pages 18–25, 2000.
13. J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.
14. X. Y. Stella. Segmentation using multiscale cues. In *Proc. CVPR*, pages I:247–254, 2004.
15. Y. Wexler, A. Fitzgibbon, and A. Zisserman. Bayesian estimation of layers from multiple images. In *Proc. ECCV*, volume 3, pages 487–501, 2002.
16. R. Zabih and V. Kolmogorov. Spatially coherent clustering using graph cuts. In *Proc. CVPR*, volume 2, pages 437–444, 2004.