Supervised Texture Detection in 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 presents a technique for texture segmentation in images. Providing a small template of a texture of interest results in the image being segmented into regions with similar properties and background (non-similar) regions. The core of the segmentation engine is based on the minimal cut/maximal flow algorithm in the graph representing an image. The main contribution lies in incorporating the template information (colour, texture) into the whole graph used for segmentation. The method brings the possibility to locate textured regions in the image having same property as the template patch and not only one-colored regions (as in much existing work). The method is supervised since the user provides a representative template of an object being searched for. The object may consist of several isolated parts. Experimental results are presented on some images from the Berkeley database.

1 Introduction

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

There are many papers dealing with automatic segmentation. We mention only the work of Shi & Malik [12] based on normalized cuts which segments the image into many non-overlapping regions. They introduced a modification of graph cuts, namely normalized graph cuts, and provided an approximate closed-form solution. However, the boundaries of detected regions often do not follow the true boundaries of the objects. The work [13] is a follow-up to [12] where the segmentation is done at various scales. The final segmentation is then glued together from partial ones.

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 [11,14,1], or *ii)* specifying the foreground object properties [3,10,2].

* 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).

A. Gagalowicz and W. Philips (Eds.): CAIP 2005, LNCS 3691, pp. 441–448, 2005.

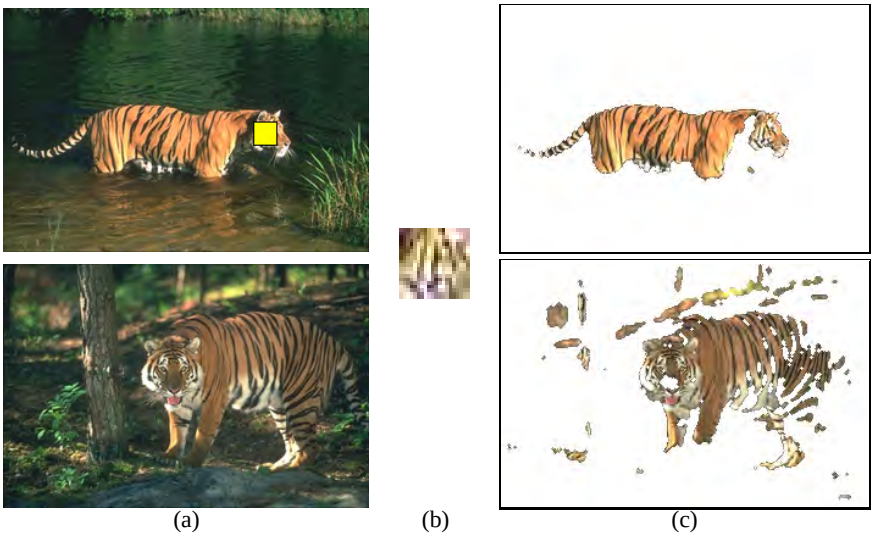


Fig. 1. Supervised segmentation. (a) Original image. Top image shows marked place from which the template was cut. (b) The enlarged template patch. (c) binary segmentation with masked original image.

In this paper we concentrate on an easier task than fully automatic segmentation. We constrain the segmentation by using a small user-provided template patch. We search for the segments of the image coherent in terms of colour with the provided template. The texture of an input image is taken into account to correctly detect boundaries of textured regions.

The proposed technique could be useful for segmentation and for detecting the objects in images with a characteristic a priori known property defined through a template patch. Fig. 1 (top row) shows how a tiger can be detected in the image using a small template patch from the same image. The same template patch can be used to detect the tiger in another image even though lighting conditions are slightly different, see Fig. 1 (bottom row).

In this paper 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 the 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. We improved the method in [8] to cope with colour and texture images. However, both seeds for background and foreground objects were still needed. In this work we avoid the need of background seeds and only seeds for the foreground object need to be specified.

In [15] 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 [5]. In our work we follow a similar strategy. However, we define the neighborhood relation through brightness, colour and texture gradients introduced in [6,7].

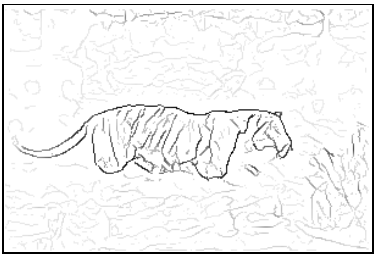


Fig. 2. Combined boundary probability using colour+texture gradient of the tiger image. Black points stand for high, white for low boundary probability.

In the paper [10] a similar idea to ours has been treated. In principle, the result is the same, but the strategy to obtain it differs. The boundary of a textured foreground object is achieved by minimization (through the evolution of the region contour) of energies inside and outside the region. The Geodetic Active Region framework is used to propagate the region contour. However, the texture information for the foreground has to be specified by the user. In [9] the user interaction is omitted. At first number of regions is estimated by fitting a mixture of Gaussians on intensity histogram and then used to drive the region evolution. However, such technique cannot be used for textured images. One textured region can be composed of many colours and therefore Gaussian components say nothing about number of dominant textures.

The main contribution of this paper lies in incorporating the information included in the template patch into the graph representing the image, leading to a reasonable binary image segmentation. Our method does not need seeds for both foreground and the background as in [3,8]. Only some representative template patch of the object being searched for is required, see Fig. 1. Moreover, texture information is taken into account.

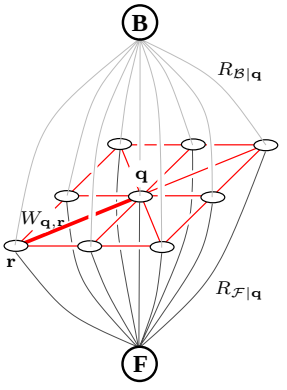
The structure of the paper is as follows. First, segmentation based on the graph cut algorithm is outlined together with energy functions. Second, non-parametric computation of probabilities of points being foreground/background through histograms and incorporating template patch information is described. Finally, the results and summary conclude the work.

2 Segmentation

We used a segmentation technique based on the interactive graph cut method first introduced in [3]. There exists a very efficient algorithm for finding min-cut/max-flow in a graph [4]. At first we very briefly outline the boundary detection and then focus in more detail on the construction of the graph representing the image.

2.1 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



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

Fig. 3. Left: Graph representation for 9 pixel image. Right: Table defining the costs of graph edges. λ is a constant described in the text.

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 like in work [6,7]. In this paper we use as a cue the brightness, colour, and texture gradients introduced in [6,7] to produce the combined boundary probability, see Fig. 2. For more details see also [8].

2.2 Graph Representing the Image

We introduced new penalties on edges in a graph based on a combined boundary probability image. The general framework for building the graph is depicted in Fig. 3. The graph is shown for a 9 pixel image and an 8-point neighborhood $\mathcal{N}$. For general images, the graph has as many nodes as pixels plus two extra nodes labeled F, B. In addition, the pixel neighborhood is larger.

Neighboring Pixel Relation

The edge weights of neighborhood $\mathcal{N}$ are encoded in the matrix $W_{\mathbf{q},\mathbf{r}}$,which is not necessarily symmetric. 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.

The neighborhood penalty of two pixels is defined as follows

$$W_{\mathbf{q},\mathbf{r}} = \left(e^{-\frac{g(\mathbf{q},\mathbf{r})^2}{\sigma_2}} \right)^2 , \tag{1}$$

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

$$g(\mathbf{q},\mathbf{r}) = p_b(\mathbf{q}) + \max_{\mathbf{s}\in\mathcal{L}_{\mathbf{q},\mathbf{r}}} p_b(\mathbf{s}) , \tag{2}$$

where $p_b(\mathbf{q})$ is the combined boundary probability mentioned in Sec. 2.1 and

$$\mathcal{L}_{\mathbf{q},\mathbf{r}} = \{\mathbf{x} \in \mathbb{R}^2 \colon \mathbf{x} = \mathbf{q} + k(\mathbf{r} - \mathbf{q}), k \in (0, 1\rangle\}$$

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. (2) follows the idea that there is a large weight if the line connecting two points crosses the 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 is zero.

Foreground / Background Nodes

Each node in the graph is connected to the two extra nodes F, B. This allows the incorporation of the information provided by the template and a penalty for each pixel being foreground or background to be set.

The regional penalty of a point as being foreground $\mathcal{F}$ or background $\mathcal{B}$ is defined as follows

$$\begin{aligned} R_{\mathcal{F}|\mathbf{q}} &= -\ln p(\mathcal{B}|\mathbf{c_q}) \\ R_{\mathcal{B}|\mathbf{q}} &= -\ln p(\mathcal{F}|\mathbf{c_q}), \end{aligned} \tag{3}$$

where $\mathbf{c_q} = (c_L,\ c_a,\ c_b)^\top$ stands for a vector in $\mathbb{R}^3$ of L*a*b* values at the pixel $\mathbf{q}$. We use the L*a*b* colour space as this results in better performance. This color space is approximately perceptually uniform and Euclidean distances in this space are perceptually meaningful. To compute the posterior probabilities in Eq. (3) we used Bayes' theorem as follows

$$p(\mathcal{B}|\mathbf{c_q}) = \frac{p(\mathbf{c_q}|\mathcal{B})\, p(\mathcal{B})}{p(\mathbf{c_q})} = \frac{p(\mathbf{c_q}|\mathcal{B})\, p(\mathcal{B})}{p(\mathcal{B})\, p(\mathbf{c_q}|\mathcal{B}) + p(\mathcal{F})\, p(\mathbf{c_q}|\mathcal{F})}. \tag{4}$$

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. (4) reduces to

$$p(\mathcal{B}|\mathbf{c_q}) = \frac{p(\mathbf{c_q}|\mathcal{B})}{p(\mathbf{c_q}|\mathcal{B}) + p(\mathbf{c_q}|\mathcal{F})}, \tag{5}$$

where the prior probabilities are

$$p(\mathbf{c_q}|\mathcal{F}) = f^L_{c_L} \cdot f^a_{c_a} \cdot f^b_{c_b}, \quad \text{and} \qquad p(\mathbf{c_q}|\mathcal{B}) = b^L_{c_L} \cdot b^a_{c_a} \cdot b^b_{c_b},$$

where $f_i^{\{L,a,b\}}$, resp. $b_i^{\{L,a,b\}}$, represents the foreground, resp. the background histogram of each colour channel separately at the ith bin.

All histogram channels are smoothed using one-dimensional Gaussians, i.e. $\bar{f}_i = \frac{1}{G}\sum_{j=1}^{N} f_j e^{-\frac{(j-i)^2}{2\sigma^2}}$, where G is the normalization factor enforcing $\sum_{i=1}^{N} \bar{f}_i = 1$. In our case, the number of histogram bins, $N = 64$. We used $\sigma = 1$ since experiments showed that it is a reasonable value. λ from the table in Fig. 3 was set to 1000.

In an implementation one should take into account the possibility of a zero value of $p(\mathcal{B}|\mathbf{c_q})$ in Eq. (3) and thus avoid an overflow. In such a case $R_{\mathcal{F}|\mathbf{q}} = K$, where K is some "big" number (we use 10000).

The foreground histogram is computed from all pixels in the template patch. To compute the background histogram is a little bit tricky. We know a priori neither the colours nor a template patch of the background. We suggest to compute the background histogram from all image pixels. The basic idea behind this is the assumption that the histogram computed from all points includes information on all colours (the background and the foreground) in the image. Therefore, since $\sum_{i=1}^{N} \bar{b}_i = 1$, the probability $p(\mathbf{c_q}|\mathcal{B})$ gives smaller values than $p(\mathbf{c_q}|\mathcal{F})$ for the colours present in the template. Thus, points more similar to the template are assigned in the graph more strongly to the foreground than to the background node.

3 Experiments

The segmentation method was implemented in MATLAB. Some of the most time consuming operations (such as 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 [6].

The most time consuming part of the segmentation process is creating the weight matrix W. It takes 50 seconds on a 250×375 image running on a Pentium 4@2.8 GHz. The implementation of the texture gradient in C would dramatically speed up the computation time. Once the graph is built, finding the min-cut takes 2 – 8 seconds.

In all our experiments we used images from the Berkeley database. We marked a small "representative" part of the image and used it for further image segmentation of the image. See Fig. 4 for the results. From the results it can be seen that very good segmentation can be obtained even though only the colour histogram of the template patch is taken into account.

It is also possible to apply the template patch obtained from one image for segmenting another one. In the case depicted in Fig. 1 small tiger patch encoding the tiger's colours obtained from one image is used for finding the tiger in another image. It can be seen that most of the tiger's body was captured but also some pixels belonging to the background were segmented. Such "non-tiger" regions could be pruned using some further procedure, which is not discussed in this paper.

It opens new possibilities for the use of the method, e.g., for image retrieval applications. Since some representative image template is available, images from large databases coherent in colour and texture can be found.

4 Discussion

In the method presented in this paper, only colour histograms are used for description of the texture of the template. Hence there are some limitations. First of all there is a problem in the change of lighting conditions and the presence of shadows. Every

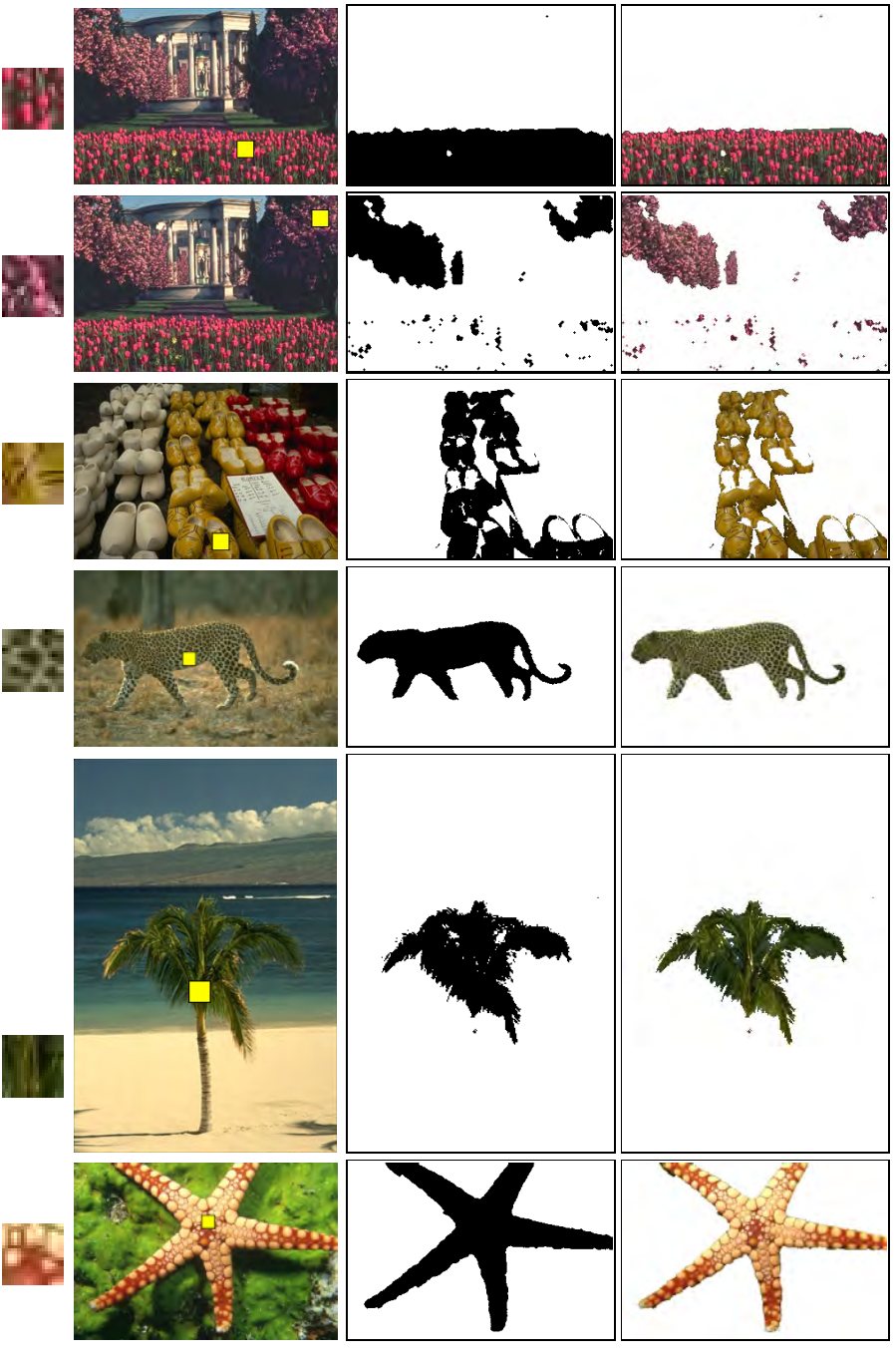


Fig. 4. Results (we recommend to see a colour version of this Figure). 1st column: enlarged image template patch. 2nd column: input image with marked area used as the template. 3rd column: binary segmentation. 4th column: segmentation with masked original image.

image is usually captured under different conditions and to find one universal template capturing all such changes is almost impossible.

However, other properties of the texture, e.g. inner texture structure, topology of texture elements, should be taken into account to improve the robustness of the algorithm to the change of lighting conditions and shadows. We are currently working on this.

5 Conclusion

We suggested a method for supervised texture segmentation in images. The method is based on finding the min-cut/max-flow in a graph representing the image to be segmented. The paper described how to set the weights of graph edges to handle the information present in a small representative template patch provided by the user. We proposed a new strategy to avoid the need for a background template or for a priori information of the background. Experiments presented on some images from the Berkeley database show that the method gives reasonable results.

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. Prez, 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. V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *PAMI*, 26(2):147–159, 2004.
6. J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *IJCV*, 43(1):7–27, 2001.
7. 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.
8. B. Micusik and A. Hanbury. Steerable semi-automatic segmentation of textured images. In *Proc. Scandinavian Conference on Image Analysis (SCIA)*, 2005.
9. 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.
10. N. Paragios and R. Deriche. Geodesic active regions and level set methods for supervised texture segmentation. *IJCV*, 46(3):223–247, 2002.
11. M. Ruzon and C. Tomasi. Alpha estimation in natural images. In *Proc. CVPR*, volume 1, pages 18–25, 2000.
12. J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.
13. X. Y. Stella. Segmentation using multiscale cues. In *Proc. CVPR*, volume 1, pages 247–254, 2004.
14. Y. Wexler, A. Fitzgibbon, and A. Zisserman. Bayesian estimation of layers from multiple images. In *Proc. ECCV*, volume 3, pages 487–501, 2002.
15. R. Zabih and V. Kolmogorov. Spatially coherent clustering using graph cuts. In *Proc. CVPR*, volume 2, pages 437–444, 2004.