Automatic Image Segmentation by Positioning a Seed*

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. We present a method that automatically partitions a single image into non-overlapping regions coherent in texture and colour. An assumption that each textured or coloured region can be represented by a small template, called the seed, is used. Positioning of the seed across the input image gives many possible sub-segmentations of the image having same texture and colour property as the pixels behind the seed. A probability map constructed during the sub-segmentations helps to assign each pixel to just one most probable region and produce the final pyramid representing various detailed segmentations at each level. Each sub-segmentation is obtained as the min-cut/max-flow in the graph built from the image and the seed. One segment may consist of several isolated parts. Compared to other methods our approach does not need a learning process or a priori information about the textures in the image. Performance of the method is evaluated on images from the Berkeley database.

1 Introduction

Image segmentation can be viewed as a partitioning of an image into regions having some similar properties, e.g. colour, texture, shape, etc, or as a partitioning of the image into semantically meaningful parts (as people do). A common problem is that it is difficult to objectively measure the goodness of a segmentation produced for such a task. Obtaining absolute ground truth is almost impossible since different people produce different manual segmentations of the same images [1].

Recently, a method combining image segmentation, the detection of faces, and the detection and reading of text in an integrated framework has appeared [2]. It is one of the first attempts to look at segmentation as a knowledge-driven task. At the beginning of the whole face/text recognition task a pre-segmentation of the image is performed which is then iteratively improved by the recognition results. It turns out that the knowledge-based approach using good initial segmentation leads to a reasonable result towards recognition of the objects in images. Similarly, in [3] it is shown that the image segmentation is an important first step in automatic annotation of pictures.

In this paper we concentrate on finding an initial segmentation without any a priori knowledge such as an object database. The image is split automatically into regions

* 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. Leonardis, H. Bischof, and A. Prinz (Eds.): ECCV 2006, Part II, LNCS 3952, pp. 468–480, 2006.

Fig. 1. Automatic segmentation of the zebra image shown at the left. The three images on the right show three dominant textures as three different regions produced by the proposed method.

having similar properties in terms of colour and texture. See Fig. 1, where zebras were segmented due to different texture and colour to the grass background. This should be useful in a cognitive vision system leading towards the understanding of an image, as in [2, 3]. As psychophysics experiments have shown [4], at the beginning of the human procedure leading to scene understanding, some pre-segmentation using boundaries and regions is performed as well. Finally, humans use a huge object database in their brains to tune the segmentation. Usually, even with large occlusions, strong shadows and geometric distortions, humans still are able to recognize objects correctly.

There are many papers dealing with automatic segmentation. We have to mention the well known work of Shi & Malik [5] based on normalized cuts which segments an image into 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 [6] is a follow-up to [5] where the segmentation is improved by doing it at various scales.

The normalized cuts method has often been used with success in combination with methods computing pixel neighborhood relations through brightness, colour and texture cues [7, 8, 9, 10]. See results [11] showing what automatic segmentation without knowledge database using affinity functions [8] which were fed to an eigensolver to cluster the image can achieve. In our experiments we used the same image dataset [12] to easily compare the results.

There is another direction in image segmentation by using Level Set Methods [13, 14]. The boundary of a textured foreground object is obtained by minimization (through the evolution of the region contour) of energies inside and outside the region.

The main contribution of this paper lies in showing how a small image patch can be used to automatically drive the image segmentation based on graph cuts resulting in colour- and texture-coherent non-overlapping regions. Moreover, a new illumination invariant similarity measure between histograms is designed. For finding min-cut/max-flow in the graph we applied the algorithm [15] used for the user-driven image segmentation for grayscale non-textured images [15, 16, 17] augmented to colour and textured images in [18].

The proposed method works very well for images containing strong textures like natural images, see Fig. 1. Compared to other methods our approach does not need a learning process [8] or a priori information about the textures in the image [13]. The method positions a circular patch, called the seed, to detect the whole region having the same properties as the area covered by the seed. Many sub-segmentations produced during the positioning of the seed are then merged together based on proposed similarity

measures. To obtain semantically correct regions composed often of many segments with different textures and colours some knowledge-based method would have to be applied which, however, is out of the scope of this paper.

A similar idea for establishing seeds at salient points based on a spectral embedding technique and min-cut in the graph appeared in [19]. However, we provide another more intuitive solution to this problem.

The structure of the paper is as follows. The segmentation method is first explained for one seed in Sec. 2 and then for multiple seeds together with combining and merging partial segmentations yielding the final segmentation pyramid in Sec. 3, outlined in steps in Sec. 4. Finally an experimental evaluation and summary conclude the paper.

2 One Seed Segmentation

We use a seed segmentation technique [18] taking into account colour and texture based on the interactive graph cut method [15]. The core of the segmentation method is based on an efficient algorithm [16] for finding the min-cut/max-flow in a graph. At first we very briefly outline the boundary detection and then the construction and segmentation of the graph representing an image.

2.1 Boundary Detection

Our main emphasis is put on boundaries at the changes of different textured regions and not local changes inside a single texture. However, there are usually large responses of edge detectors inside textures. Therefore, in this paper we use as a cue the colour and texture gradients introduced in [7, 9] to produce the *combined boundary probability* image, see Fig. 5(b).

2.2 Graph Representing the Image

The general framework for building the graph is depicted in Fig. 2 (left). The graph is shown here for a 9 pixel image and an 8-point neighborhood $\mathcal{N}$. In general, the graph has as many nodes as pixels plus two extra nodes labeled F, B. In addition, the pixel neighborhood is larger, e.g. we use a window of size 21×21 pixels.

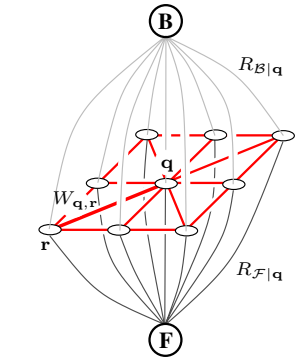
The neighborhood penalty between two pixels is defined as follows

$$W_{\mathbf{q},\mathbf{r}} = \left(e^{-\frac{g(\mathbf{q},\mathbf{r})^2}{\sigma_2}} \right)^2, \qquad g(\mathbf{q},\mathbf{r}) = p_b(\mathbf{q}) + \max_{\mathbf{s}\in\mathcal{L}_{\mathbf{q},\mathbf{r}}} p_b(\mathbf{s})\,, \tag{1}$$

where σ_2 is a parameter (we used $\sigma_2 = 0.08$ in all our experiments), $p_b(\mathbf{q})$ is the combined boundary probability (Sec. 2.1) at point $\mathbf{q}$ 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 discretized line from the point $\mathbf{q}$ (exclusive) to the point $\mathbf{r}$ (inclusive).

Each node in the graph is connected to the two extra nodes F, B. This allows the incorporation of the information provided by the seed and a penalty for each pixel being foreground or background to be set. The penalty of a point as being foreground $\mathcal{F}$ or background $\mathcal{B}$ is defined as follows

$$R_{\mathcal{F}|\mathbf{q}} = -\ln p(\mathcal{B}|\mathbf{c}_\mathbf{q}), \qquad R_{\mathcal{B}|\mathbf{q}} = -\ln p(\mathcal{F}|\mathbf{c}_\mathbf{q}), \tag{2}$$



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

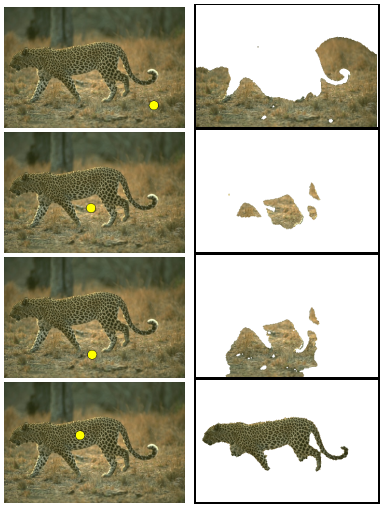


Fig. 2. Left: Graph representation for a 9 pixel image and a table defining the costs of graph edges. Symbols are explained in the text. Right: Four binary image segmentations using various positions of the seed.

where $\mathbf{c_q} = (c_L,\ c_a,\ c_b)^\top$ is a vector in $\mathbb{R}^3$ of CIELAB values at the pixel $\mathbf{q}$. The CIELAB colour space has the advantage of being approximately perceptually uniform. Furthermore, Euclidean distances in this space are perceptually meaningful as they correspond to colour differences perceived by the human eye. Another reason for the good performance of this space could be that in calculating the colour probabilities below, we make the assumption that the three colour channels are statistically independent. This assumption is better in the CIELAB space than in the RGB space. The posterior probabilities are computed as

$$p(\mathcal{B}|\mathbf{c_q}) = \frac{p(\mathbf{c_q}|\mathcal{B})}{p(\mathbf{c_q}|\mathcal{B}) + p(\mathbf{c_q}|\mathcal{F})}, \tag{3}$$

where the prior probabilities are

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

and $f^{\{L,a,b\}}(i)$, resp. $b^{\{L,a,b\}}(i)$, represents the foreground, resp. the background histogram of each colour channel separately at the ith bin smoothed by a Gaussian kernel. We used 64 bins. The foreground histograms $f^{\{L,a,b\}}$ are computed from all pixels behind the seed. The background histograms $b^{\{L,a,b\}}$ are computed from all pixels in the image. See [18] for more details. λ in the table in Fig. 2 controls the importance of foreground/background penalties against colour+texture penalties and was set to 1000.

After the graph is built the min-cut/max-flow splitting the graph and also the image into two regions is found by the algorithm [16].

See segmentations resulting from various seed positions in Fig. 2 (right). It can be seen that segmented foreground region has similar properties to the pixels behind the

seed. Due to illumination changes, shadows and perspective distortion changing the resolution of textures, the whole texture region is usually not marked as one region. However, the segmented regions representing the same texture overlap which we use in the procedure described in the next section to merge them and to build a probability map yielding the segmentation.

3 Multiple Seed Segmentation

3.1 Seed Positioning

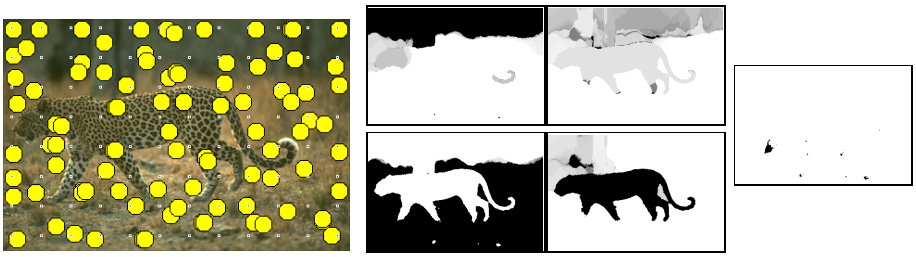
Each seed position gives one binary segmentation of the image, see Fig. 2(right). To obtain image segmentation we move the seed across the image as follows.

A regular grid of initial seed positions is created, marked as black dots on small white patches in Fig. 3(a). Using seeds at regular grid positions would segment two textured regions as one segment. Since we want to find segments with a constant inner structure we avoid cases where the seed crosses a strong response in the combined boundary probability map in Fig. 5(b). Therefore we create a local neighborhood around each initial position in the grid and the position of the seed which minimizes the sum of values of pixels behind the seed in the combined probability map is looked for, i.e.

$$\mathbf{u}^* = \underset{\mathbf{u}\in\mathcal{A}}{\operatorname{argmin}} \sum_{\mathbf{v}\in\mathcal{S}_\mathbf{u}} p_b(\mathbf{v}), \tag{4}$$

where $\mathbf{u}$ is a 2 element vector with $(x, y)^\top$ image coordinates, $\mathcal{S}_\mathbf{u}$ is the seed (in our case circular) area centered at the point $\mathbf{u}$ and $\mathcal{A}$ is a neighborhood rectangle around the initial grid point. The neighborhood rectangles should not overlap to avoid the case of identical seed positions having different initial points. We find the minimum in Eq. (4) by brute force, i.e. the error is evaluated at all possible positions of the seed in the neighborhood $\mathcal{A}$ because of low computational demand.

For each initial grid position, one $\mathbf{u}^*$ is found and the segmentation method described in Sec. 2 is applied using a seed positioned at $\mathbf{u}^*$ to obtain a binary sub-segmentation. The positions $\mathbf{u}^*$ of the seeds for the leopard image are shown in Fig. 3(a).



(a) (b) (c)

Fig. 3. (a) The input image with automatically positioned seeds. (b) Probability maps for four possible segments. Black corresponds to the higher probability, white to the lowest one. (c) Unassigned pixels.

3.2 Combining Partial Segmentations

The sub-segmentations corresponding to the seeds are grouped together w.r.t. the size of the mutual common area with other sub-segmentations. At the beginning of moving the seed an empty list of potential segments is created. After the first run (first position of the seed) the sub-segmentation is assigned to the first segment in the list. After each consecutive run the actual sub-segmentation is compared to segments already stored in the list. If there is any segment in the list overlapping with a specified fraction (we use 80%) of pixels then the sub-segmentation is *summed* to this segment. Otherwise a new segment in the list is created.

Summing the sub-segmentations produces the probability with which each pixel belongs to each of the possible segments. The sum of values of pixels lying at the same position in different segments in the list is used for normalization to get the value range from 0 to 1. Fig. 3(b) shows an example of a four segment list obtained by applying segmentations using seeds depicted in Fig. 3(a). There may still remain pixels which were not assigned to any segment, see Fig. 3(c), which are treated in the merging stage described later.

3.3 From Probability Map to Segments

The probability map constructed in the previous sub-section can be used to obtain the a priori probability of each possible segment. Assuming that each segment is equally important and no penalizations are applied, the decision following Bayes theorem leads to choosing for each pixel the segment which has the highest support by sub-segmentations, i.e. has highest a priori probability. For example, the tail of the leopard is present in three segments, see Fig. 3(b). However, in the segment containing the whole leopard the pixels corresponding to the tail have the highest probability to be assigned to this segment. See Fig. 4 for the result. The list of segments $\mathcal{L}$ is represented by binary matrices $\mathtt{L}_i$, i.e.

$$\mathcal{L} = \{\mathtt{L}_i \in \{0,1\}^{n \times m} : 0 \leq i \leq S\},$$

where S is the number of segments. The matrix $\mathtt{L}_0$ stands for the segment containing unassigned pixels.

For the leopard image after this stage we could be satisfied since the segmentation captures the main regions. One possible region (top right in Fig. 3(b)) disappeared as no pixels remained assigned to this segment after incorporating probabilities. However, the non-overlapping segments having similar properties can sometimes be split due to illumination changes. To observe this, look at the grass or the bear head in the bear image

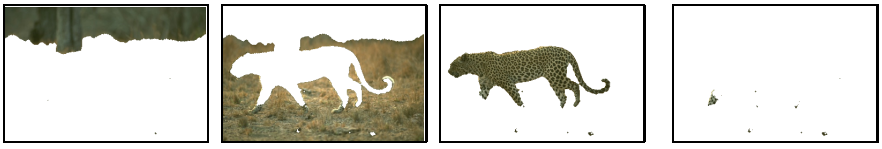
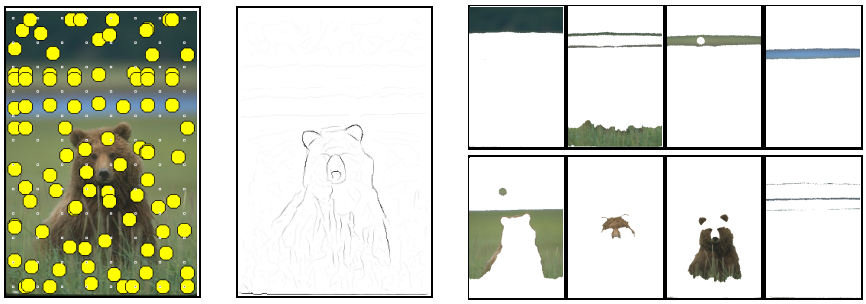


Fig. 4. Segmentation after assigning the most probable segment to each pixel. The rightmost image corresponds to unassigned pixels.



(a) (b) (c)

Fig. 5. (a) The bear image with automatically positioned seeds. (b) Combined boundary probability image. (c) Possible segments. The last one corresponds to unassigned pixels.

segmentation in Fig. 5. Therefore, we incorporate a shadow-invariant colour space and merge similar segments into one using a newly designed similarity measure described in the following subsections.

3.4 Elimination of Unassigned Segments

We convert an input image into the $c_1c_2c_3$ illumination invariant colour space [20]. Comparison and evaluation of various colour models in the sense of their invariance can be found in [20]. The conversion from RGB to $c_1c_2c_3$ colour space is done as follows

$$c_1 = \arctan \frac{R}{\max\{G,B\}}, \quad c_2 = \arctan \frac{G}{\max\{R,B\}}, \quad c_3 = \arctan \frac{B}{\max\{R,G\}}.$$

We compute colour histograms $h_i^{\{c_1,c_2,c_3\}}$ from pixels marked in segment $\mathtt{L}_i$ by 1's for $1 \leq i \leq S$. We used 64 bins and smoothed the histograms by a Gaussian kernel.

We label an unassigned segment stored in the binary matrix $\mathtt{L}_0$ to separate all regions in this image. For each region $\mathcal{R}_j$ in the segment $\mathtt{L}_0$, if its area is larger than some threshold (we use 200 pixels), the new segment $\mathtt{L}_{S++}$ is added into the list of all segments $\mathcal{L}$. Otherwise, if the area is below the threshold, the region $\mathcal{R}_j$ is assigned to the most probable segment i^* in the list $\mathcal{L}$ w.r.t. to the following criterion

$$i^*(j) = \underset{1\leq i\leq S}{\operatorname{argmax}} \sum_{\mathbf{u}\in\mathcal{R}_j} h_i^{c_1}(\mathtt{I}(\mathbf{u})_{c_1}) \cdot h_i^{c_2}(\mathtt{I}(\mathbf{u})_{c_2}) \cdot h_i^{c_3}(\mathtt{I}(\mathbf{u})_{c_3}), \tag{5}$$

where $\mathtt{I}(\mathbf{u})_{\{c_1,c_2,c_3\}}$ are c_1, c_2, c_3 values of an image point at the position $\mathbf{u}$. By this step all pixels/regions in the unassigned segment $\mathtt{L}_0$ are eliminated, however, the number of segments in the list $\mathcal{L}$ can increase.

3.5 Merging Segments

We observed that the change of illumination on the same surface does not change the shape of the histograms, however, it causes their mutual shift. This motivated us to

design a new illumination invariant similarity function between histograms based on evaluating the shift.

At first, compute the cross-correlation between histograms of segments for each colour channel separately and find the maximum values of cross-correlation in some range $\langle t_1, t_2 \rangle$, i.e.

$$\mathbf{r}(i,j) = \begin{pmatrix} \operatorname{argmax}_{t_1 \le t \le t_2} (h_i^{c_1} \star h_j^{c_1})(t) \\ \operatorname{argmax}_{t_1 \le t \le t_2} (h_i^{c_2} \star h_j^{c_2})(t) \\ \operatorname{argmax}_{t_1 \le t \le t_2} (h_i^{c_3} \star h_j^{c_3})(t) \end{pmatrix}, \tag{6}$$

where $\star$ stands for cross-correlation. We show in Fig. 6 the cross-correlation of third segment histograms with each of the other segments, i.e. $(h_3^{c_{\{1,2,3\}}} \star h_j^{c_{\{1,2,3\}}})(t)$, for the segments shown in Fig. 5(c). As can be seen the cross-correlations have single maxima which can easily be detected. If there is no peak inside the interval bounded by t_1, t_2, the distance is set to Inf. We use $t_2 = -t_1 = 20$. The interval should be reasonably narrow since comparison of the same colours affected by shadows yields only small displacement of the maxima. In contrast, comparison of different colours yields more significant displacement and the distance between maxima is meaningless.

Let three elements of $\mathbf{r}(i,j)$ be sorted in a vector $\mathbf{s} = (s_1, s_2, s_3)^\top$ such that $s_1 \le s_2 \le s_3$. The squared distance of two histograms i, j is then evaluated as

$$d(i,j) = (s_1 - s_2)^2 + (s_3 - s_2)^2. \tag{7}$$

The histogram distance in Eq. (7) computed for all pairs of segments is used for finding most similar segment(s) in the list $\mathcal{L}$. The segments which mutually match to each other are merged together if the distance is below some threshold d_{thr}. The level of merging can be controlled by this threshold. Depending on the value various levels in the final segmentation pyramid are created, see for example the three-level pyramid in Fig. 7. In this case d_{thr} was increasing from 10 to 200 while three levels were obtained.

From Fig. 6 it is evident that the grass segment (third segment in Fig. 5(c)) is most similar to other green segments. The same happens to the bear's head which is at first

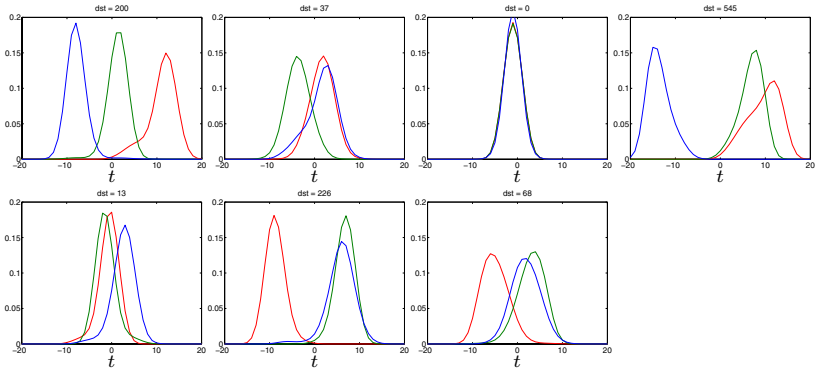


Fig. 6. Histogram cross-correlation $(h_3^{c_{\{1,2,3\}}} \star h_j^{c_{\{1,2,3\}}})(t)$ for $j = 1..7$. Red, green, blue colour of the curves in each graph corresponds to the c_1, c_2, c_3 colour channel respectively.

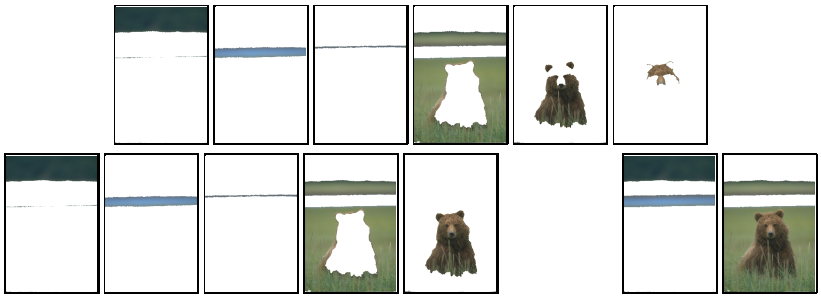


Fig. 7. Three-level pyramid of the bear image. Top row: First pyramid level with six segments. Bottom row: Second level (five segments on the left) and third level (two rightmost segments).

divided into two parts in Fig. 5(c), however, at some level in the pyramid is merged together.

4 Algorithm

We shortly summarize all the steps leading to the final single image segmentation:

1. Convert the image from the RGB colour space to the CIELAB space.
2. Compute the combined boundary gradient based on [7, 9] of the image.
3. Make a regular initial grid of seeds. For each initial seed position find a new optimal position, Sec. 3.1, and compute a binary segmentation based on the min-cut/max-flow in the graph, Sec. 2.
4. Combine segmentations yielding a probability map, Sec. 3.2, and create a list of segments $\mathcal{L}$, Sec. 3.3.
5. Eliminate unassigned pixels, Sec. 3.4, and merge similar segments based on the illumination invariant similarity measure described in Sec. 3.5.
6. Depending on the chosen distance threshold d_{thr} in the similarity measure, the degree of segmentation coarseness is controlled and the final segmentation pyramid is obtained.

5 Experimental Evaluation

To benchmark the results of the algorithms, we made use of the Berkeley segmentation benchmark described in [1]. Two measures of the difference between two segmentations S_1 and S_2 are introduced in this paper, the Global and Local Consistency Errors (GCE and LCE). As the GCE is a more demanding measure, we make use of only this measure. There are other possibilities for benchmarking such as to use precision/recall curves as in [19, 9].

We used the 200 colour images in the test group of the Berkeley Segmentation Dataset [12] as well as the corresponding human segmentations. For each of the images, at least 5 segmentations produced by different people are available. For each image, the GCE of the segmentation produced by the tested algorithm with respect to each of the

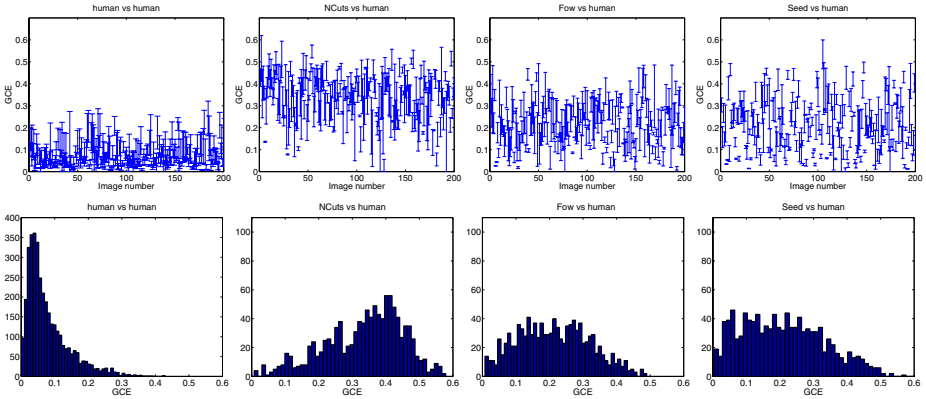


Fig. 8. Global Consistency Error (GCE) for human, normalized cuts (ncuts), Fowlkes et al. [8] (fow) and our proposed method (seed) (from left). The top row shows variance of the GCE for each image in the dataset. The bottom row shows the histogram of the GCE.

available human segmentations for that image was calculated. The mean of these values gives the mean GCE per image, which was plotted in a histogram, see Fig. 8. The global GCE was calculated as the mean of these 200 mean GCE values.

We compared human segmentations to each other and then with the normalized cuts algorithm (ncuts) [5], Fowlkes et al. algorithm (fow) [8] and our seed algorithm (seed). Comparison of human vs. human produces a very low GCE value which indicates the consistency of the human segmentations. The "ncuts" and "fow" methods were applied to the same combined boundary images as we used, mentioned in Sec. 2.1. Using the same boundary gradient implies that the performance of the various methods is compared using the same starting condition.

The implementation of the "ncuts" used (provided on the authors' web page) requires that the number of regions required be passed as a parameter. We used 5 as the average number of segments per image for our seed segmentation was 4.3. There is a version of the "ncuts" which determines the number of regions automatically [7], but we currently have no implementation of it. The segmentations for the "fow" method were provided directly by the author. In this segmentation, the average number of segments was 13. See Tab. 1 for the results.

Table 1. Comparison of the methods. The first column contains the acronyms of the methods. The second column corresponds to the average number of segments per image. The third column shows the mean GCE error over all segmentations.

method	**# of reg**	**GCE**
hum	17	0.080
seed	4	0.209
fow	13	0.214
ncuts	5	0.336

Usually as the number of regions per image grows it appears that the images become more over-segmented. As is mentioned in [1] the GCE measure does not penalize an over-segmentation. Our method and the "fow" method produce comparable GCE, however, the average number of segments of our method is less, approximately one third. In some cases it means that our method does not split coherent regions.

Our 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 the online available C++ implementations of the min-cut algorithm [16] and some MATLAB code for colour and texture gradient computation [7].

The method is relatively slow, for one 375x250 image with 96 seeds it needs on average 15 minutes on a Pentium 4@2.8 GHz. However, the computation can easily be parallelized as each sub-segmentation can be done independently on many computers. The building of the weight matrix `W` representing the graph (which is done only once per image) needs approximately 50 seconds. Once the graph is built, finding the min-cut for one seed position takes 2 – 10 seconds.

Fig. 9. Some segmentation results on images from the Berkeley dataset

You may look at the results of the "fow" method [11] and our method[1] to visually compare their performance. In general, both methods perform comparably, however, one method performs better on some images, the second one on others. This gives an option to combine the methods in some further processing to choose the better result. Some segmentations using our method can be seen in Fig. 9. The results shown here correspond to the threshold d_{thr} equal to 10. The whole pyramid was built by changing the d_{thr} from 10 to 200. Each new level of pyramid is created when the number of segments increases according to the previous level. Usually, 4 levels is the maximum.

Constants (number of histogram bins, sigmas, etc.) which appear in the text are tuned experimentally on real images to obtain reasonable performance on large data.

[1] `http://www.prip.tuwien.ac.at/Research/muscle/Images/ECCV06res`

6 Conclusion

The paper proposes a method for image segmentation into texture and colour coherent segments. The segmentation combines known algorithms for computing combined boundary gradient and for finding min-cut/max-flow in the graph. The novelty is in introducing the positioning of the seed, and collecting and merging similar segments yielding the segmentation pyramid. Moreover, an illumination invariant similarity measure is introduced.

We show that our method gives comparable results to the state-of-the-art methods based on normalized graph cuts on the Berkeley dataset. We cannot say if the proposed method outperforms existing methods since quantitative comparison of segmentations is still an open problem. However, visual comparison as well as GCE comparison indicate reasonable and useful results.

References

1. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. ICCV. (2001) 416–425
2. Tu, Z., Chen, X., Yuille, A., Zhu, S.: Image parsing: Unifying segmentation, detection, and recognition. IJCV **63**(2) (2005) 113–140 *Marr Prize*.
3. Barnard, K., Duygulu, P., de Freitas, N., Forsyth, D., Blei, D., Jordan, M.I.: Matching words and pictures. Journal of Machine Learning Research **3** (2003) 1107–1135
4. Wolfson, S., Landy, M.: Examining edge- and region-based texture analysis mechanisms. Vision Research **38**(3) (1998) 439–446
5. Shi, J., Malik, J.: Normalized cuts and image segmentation. PAMI **22**(8) (2000) 888–905
6. Stella, X.Y.: Segmentation using multiscale cues. In: Proc. CVPR. (2004) I: 247–254
7. Malik, J., Belongie, S., Leung, T., Shi, J.: Contour and texture analysis for image segmentation. IJCV **43**(1) (2001) 7–27
8. Fowlkes, C., Martin, D., Malik, J.: Learning affinity functions for image segmentation: Combining patch-based and gradient-based approaches. In: Proc. CVPR. (2003) II: 54–61
9. Martin, D.R., Fowlkes, C.C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. PAMI **26**(5) (2004) 530–549
10. Zabih, R., Kolmogorov, V.: Spatially coherent clustering using graph cuts. In: Proc. CVPR. (2004) II:437–444
11. (`http://www.cs.berkeley.edu/~fowlkes/BSE/cvpr-segs`)
12. (`http://www.cs.berkeley.edu/projects/vision/grouping/segbench`)
13. Paragios, N., Deriche, R.: Geodesic active regions and level set methods for supervised texture segmentation. IJCV **46**(3) (2002) 223–247
14. Osher, S., Paragios, N., eds.: Geometric Level Set Methods in Imaging, Vision and Graphics. Springer-Verlag (2003)
15. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In: Proc. ICCV. (2001) 105–112
16. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. PAMI **26**(9) (2004) 1124–1137
17. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? PAMI **26**(2) (2004) 147–159

18. Mičušík, B., Hanbury, A.: Supervised texture detection in images. In: Proc. Conference on Computer Analysis of Images and Patterns (CAIP). (2005) 441–448
19. Estrada, F.J., Jepson, A.D.: Quantitative evaluation of a novel image segmentation algorithm. In: Proc. CVPR. (2005) II: 1132–1139
20. Gevers, T., Smeulders, A.: Color-based object recognition. Pattern Recognition **32**(3) (1999) 453–64