Waterfall Segmentation of Complex Scenes*

Allan Hanbury[1] and Beatriz Marcotegui[2]

[1] Pattern Recognition and Image Processing Group (PRIP), Institute of Computer-Aided Automation, Vienna University of Technology, Favoritenstraße 9/1832, A-1040 Vienna, Austria
hanbury@prip.tuwien.ac.at, http://www.prip.tuwien.ac.at
[2] Centre de Morphologie Mathématique, Ecole des Mines de Paris, 35, rue Saint-Honoré, 77305 Fontainebleau cedex, France
marcoteg@cmm.ensmp.fr, http://cmm.ensmp.fr

Abstract. We present an image segmentation technique using the morphological Waterfall algorithm. Improvements in the segmentation are brought about by using improved gradients. These are based on the detection of object boundaries learnt from human segmentations introduced by Martin et al. (2004). We avoid the usual pitfall found when applying Watershed algorithms to these boundaries, namely that the boundary lines usually contain gaps, by making use of distance functions on the boundary image. Two types of distance function are used: the classic distance function and a distance function for numerical images recently introduced by Beucher (2005). Resulting segmentations are compared to human segmentations using the Berkeley segmentation benchmark. The benchmark results show that the proposed segmentation algorithm produces segmentations comparable to those produced by the Normalised Cuts algorithm.

1 Introduction

Image segmentation is often used as a first step in general object recognition in complex, natural scenes, for example in [1, 2]. The object recognition is simplified if the regions produced by the segmentation algorithm already correspond to "meaningful" objects. Nevertheless, even humans often cannot agree on the best segmentation of such a scene [3].

Many algorithms for image segmentation are available, two of the most popular being the Normalised Cuts (NCuts) [4] and the Watershed [5]. Both of these algorithms require a way of measuring the similarity (or difference) between pixels in an image. The Watershed, for example, is usually applied to some sort of gradient of an image. A particularly promising algorithm for detecting the boundaries in an image based on brightness, colour and texture cues learnt from human segmentations of an image was presented in [6], and is briefly described

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

in Section 2. Unfortunately, these boundaries are not suitable to be used as a gradient for a Watershed algorithm due to gaps in the boundary lines. In this paper, we present a solution to this problem, which is to fill the small gaps by applying a distance transform to the boundary image, as described in Section 3. An enhanced version of the Watershed algorithm, the Waterfall algorithm (Section 4), is used to segment the images. The complete algorithm is summarised in Section 5. The comparison of the Waterfall algorithm with the NCuts algorithm using the Berkeley Segmentation Benchmark is presented in Section 6.

2 Boundaries Based on Learning

We briefly review the boundaries based on learning introduced by Martin et al. [6]. They make use of brightness, colour and texture gradients to compute the boundaries. To calculate the gradients, a circular area is moved over the image. At each pixel, for a number of orientations of a line dividing the circle into two halves, the χ^2 histogram difference is evaluated for histograms of the features in the two halves. For brightness and colour, the features are the values of L^*, a^* and b^* in the CIELAB space (taken separately) and for texture, the features are 64 textons used in [6]. For each feature, the gradient is taken to be the maximum value obtained over all the orientations of the line dividing the circle. The result of this algorithm is therefore a vector of four gradient values at every pixel (3 colour and 1 texture).

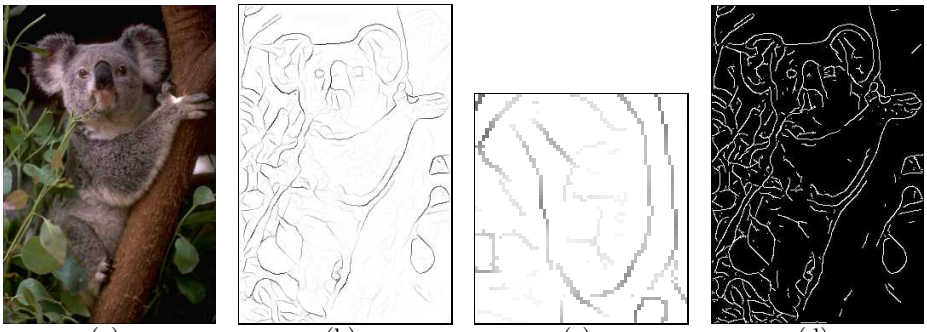
These gradients are combined to form a boundary probability by using a logistic model, where the weights for each gradient are obtained by supervised training of the model on the human segmentations. We made use of the weights provided by the authors of [6] in their software[3]. The resultant boundary probabilities are in the range $[0, 1]$. As an example, the boundaries detected in Figure 1(a) are shown in Figure 1(b).

3 Distance Functions

A common problem when attempting to segment a boundary image produced by the algorithm outlined in the previous section is the gaps in the boundary lines. These can be clearly seen in Figure 1(c), which is an enlargement of part of Figure 1(b). This results in very few local minima in the image (often only one), which makes applying Watershed based segmentation algorithms difficult. Our solution to the problem is to attempt to close the gaps by calculating a *distance function* of the boundary image.

The classic distance function takes as input a binary image. It associates with each foreground pixel the distance to the closest background pixel. See Figure 2 for an example. The maxima of the distance function (pixels represented with a hatched pattern in Figure 2(b)) mark the different particles contained in the

[3] Downloadable on the Berkeley Segmentation Benchmark page: `http://www.cs.berkeley.edu/projects/vision/grouping/segbench/`

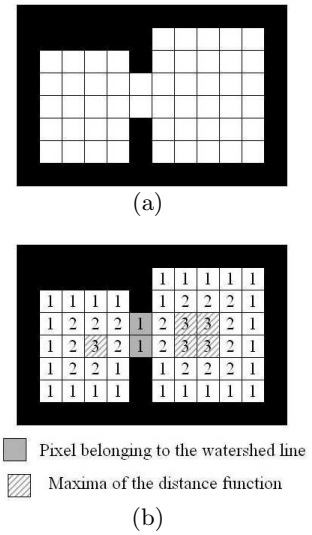


(a) (b) (c) (d)

Fig. 1. (a) An image and its (b) boundary probabilities (darker pixels indicate higher probability). (c) Detail of (b) showing the gaps in the contour. (d) Threshold of (b).

connected component. The Watershed applied to the inverse of the distance function is a well known approach for segmenting overlapping binary objects [7, 8]. In Figure 2(b), the Watershed line is represented by the grey pixels. We can see that this line correctly separates the two particles of the connected component.

If we take the boundaries detected by the Martin et al. algorithm as the background, the distance function encodes the shortest distance to each of the detected boundary lines. The value of the distance function within small gaps in the detected boundaries will therefore be lower. In the inverse of this distance function, the detected boundaries will have the maximum possible value. The lower values of the distance function in small gaps lead to higher values in the inverse, effectively closing the gaps in the topographical representation of the image used by the Watershed. Two distance functions were used: the classic distance function and the quasi-distance function.

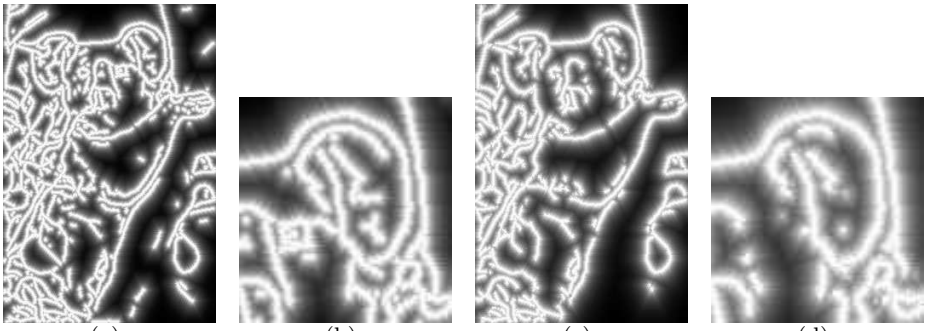


(a) (b)

Fig. 2. (a) Binary image. (b) Associated distance function.

As the classic distance function requires a binary image as input, a threshold at level t is applied to the boundary image. We used a relatively low value of $t = 0.07$ for all experiments. This was found by experiment on a number of images to be the value below which the boundaries are mostly due to noise. The threshold of Figure 1(b) is shown in Figure 1(d). The classic distance function applied to this thresholded image is shown in Figure 3(a), with a zoomed in area shown in Figure 3(b).

To avoid the necessity of choosing this threshold we also made use of the quasi-distances introduced by Beucher [9]. The quasi-distance qd of an image I



(a) (b) (c) (d)

Fig. 3. (a) Distance on the thresholded boundary image. (b) Detail of (a). (c) Quasi-distance on the boundary image (without threshold). (d) Detail of (c).

is defined as:

$$qd(x,y) = \arg\max_i(\epsilon_{i-1}(x,y) - \epsilon_i(x,y)) \tag{1}$$

where ϵ_i is the morphological erosion of size i, and (x,y) a given pixel of the image I. In other words, the quasi-distance associates with each pixel (x,y) the size i of the erosion that produces the biggest change in greylevel, among all possible sizes of erosions. Thus the quasi-distance is able to characterize the size of objects in a greylevel image without applying a threshold first. The quasi-distance function applied to the boundary image in Figure 1(b) is shown in Figure 3(c), with a zoomed in area shown in Figure 3(d).

4 Waterfall Algorithm

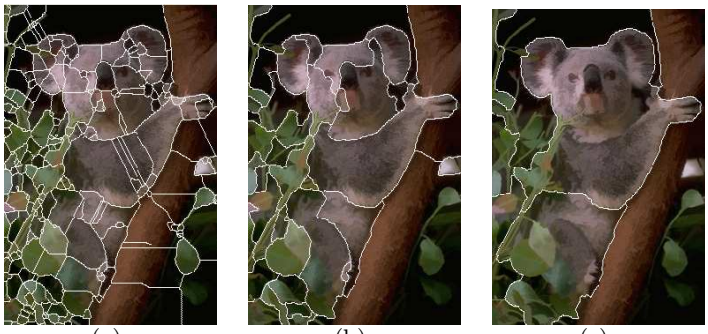
The Watershed algorithm usually leads to a strong over-segmentation of an image. The Waterfall [10] is a hierarchical approach that selects among all the contours of the Watershed those that are completely surrounded by more contrasted contours. By removing these contours, a simplified partition is obtained. The process may be iterated. At the end, a single region covering the whole image is obtained. An efficient graph-based Waterfall algorithm is presented in [11].

Examples of the Waterfall algorithm applied to the classic distance function and quasi-distance function of the detected boundary image are shown in Figures 4 and 5 respectively. In these figures, image (a) shows the result of applying the Watershed algorithm to the distance function, image (b) is the result of applying the Waterfall algorithm once (referred to as level 1 of the hierarchy) and image (c) is the result of two iterations of the Waterfall (level 2). Segmentation results on the 100 images of the Berkeley segmentation test dataset are available on the author's home page[4].

[4] `http://www.prip.tuwien.ac.at/~hanbury/ACCV06`

(a) (b) (c)

Fig. 4. (a) Watershed of distance function on the thresholded boundary probability image (level 0). (b) Waterfall level 1. (c) Waterfall level 2.



(a) (b) (c)

Fig. 5. (a) Watershed of quasi-distance on the boundary probability image (level 0). (b) Waterfall level 1. (c) Waterfall level 2.

5 Complete Segmentation Algorithm

We summarise here the algorithm used to perform the segmentation:

1. Calculate the learning-based boundaries (we used the combined colour and texture gradients [6]).
2. Calculate one of the two distance functions described in Section 3: the classic distance function on the threshold of the boundary image (abbreviated TD) or the quasi-distance function directly on the boundary image (QD).
3. Calculate the Waterfall hierarchy on the inverse of the distance function. The results of this Waterfall are referred to as "WF xD level y", where x is 'T' or 'Q', referring to the type of distance function used, and y gives the level of the Waterfall hierarchy, where level 0 is the result of the Watershed algorithm, level 1 is the first Waterfall level, etc.

6 Results and Evaluation

The results of the proposed segmentation approach are compared to those produced by the NCuts algorithm using the error measures proposed in [3].

6.1 Error Measure Definitions

To benchmark the results of the algorithms, we made use of the Berkeley segmentation benchmark [3]. Two measures of the difference between two segmentations S_1 and S_2 are introduced in [3]: the Global and Local Consistency Errors (GCE and LCE). As the GCE is a tougher measure, we make use of only this measure.

Let S_1 and S_2 be two segmentations of an image. The region $R(S, p_i)$ is the set of pixels corresponding to the region in segmentation S that contains pixel p_i. A segmentation S_1 is a *simple refinement* of S_2 if at every pixel p_i, $R(S_1, p_i) \subseteq R(S_2, p_i)$. The GCE is defined in terms of the local refinement error:

$$E(S_1, S_2, p_i) = \frac{|R(S_1, p_i) \setminus R(S_2, p_i)|}{|R(S_1, p_i)|} \tag{2}$$

where $\setminus$ denotes the set difference and $|x|$ is the cardinality of set x. As can be seen, this error measure is not symmetric. If, at pixel p_i, $R(S_1, p_i) \subseteq R(S_2, p_i)$, then $E(S_1, S_2, p_i) = 0$, but $E(S_2, S_1, p_i) > 0$. The GCE of segmentations S_1 and S_2 is defined as

$$\mathrm{GCE}(S_1, S_2) = \frac{1}{n} \min \left\{ \sum_i E(S_1, S_2, p_i), \sum_i E(S_2, S_1, p_i) \right\} \tag{3}$$

where n is the number of pixels and the sums are over all pixels. If S_1 (resp. S_2) is a simple refinement of S_2 (resp. S_1), then $\mathrm{GCE}(S_1, S_2) = 0$. As the local refinement error is not symmetrical, the minimum of the local refinement error sums calculated in both directions is taken.

We used the 100 colour test images from the Berkeley Segmentation Dataset and Benchmark as well as the corresponding human segmentations. For each of the images, at least 5 segmentations produced by different people are available. To evaluate a segmentation algorithm, it was first applied to each of the 100 images. Then, for each image, the GCE of the segmentation produced by the algorithm with respect to each of the 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. The global GCE was calculated as the mean of these 100 mean GCE values.

As the human segmentations often differ considerably, we first calculated a "best possible" GCE by comparing each human segmentation of an image to the remaining segmentations for that image. The "best possible" global GCE is 0.08 and the histogram of its distribution is shown in Figure 6(c).

For each algorithm, the mean of the number of regions produced by the segmentation algorithm for each of the 100 images was also calculated (for the

Method	GCE	Ave. #Reg.	#Reg. Agree.
WF TD level 1	0.16	51.4	24
level 2	0.23	7.8	39
WF QD level 1	0.21	28.6	46
level 2	0.22	5.6	35
N. Cuts (5 reg)	0.34	5.0	31
N. Cuts (16 reg)	0.24	16.0	63
N. Cuts (28 reg)	0.18	28.0	45
Human	0.08	16.8	-

Table 1. The Global GCE, average number of regions and region number agreement with the human segmentations for various segmentation algorithms. These are: the Waterfall algorithm (WF) operating on different types of distance function (TD and QD) for two different levels of the hierarchy, and the NCuts algorithm. The results of the human-human segmentation comparison are also shown.

human segmented images, this is 16.8). Finally, for each image, the mean $\bar{m}$ and standard deviation σ_m of the number of regions in the human segmentations is calculated. This allows the number of images for which the segmentation algorithm produces a region count lying within this range ($\bar{m} \pm \sigma_m$) to be determined (this is referred to as the *region number agreement*, shown in the rightmost column of Table 1).

6.2 Comparison of Segmentation Algorithms

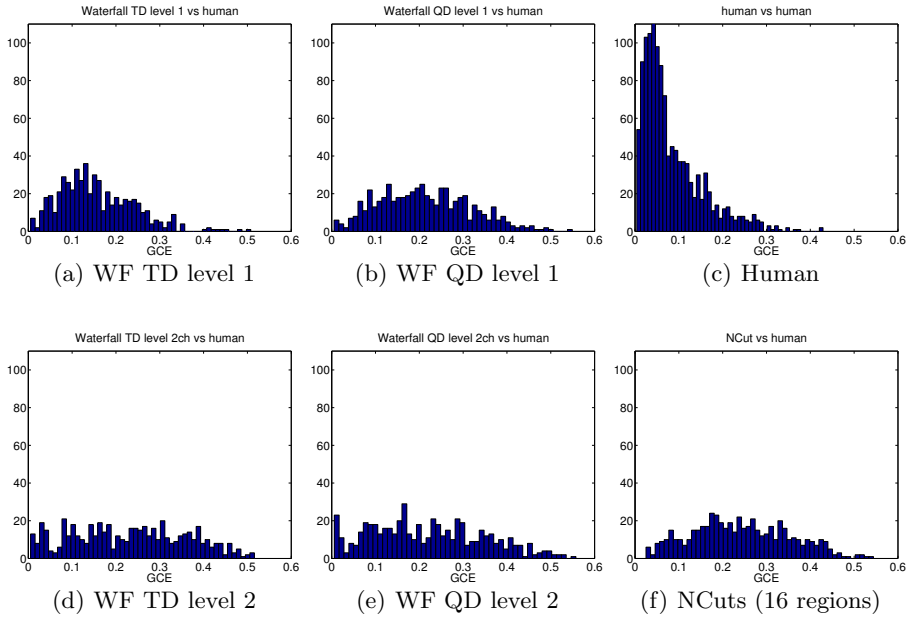
We calculated the global GCE values for levels 1 and 2 of the WF TD and the WF QD, as well as for a segmentation by the NCuts algorithm[5]. These GCE values are shown in Table 1. Histograms showing the distributions of the mean GCE values of each of the 100 images are shown in Figure 6. Note that some of the segmentations at level 2 of the Waterfall hierarchy consist of only one region. As the GCE for such a segmentation is zero, we chose to use level 1 of the hierarchy if the number of regions in level 2 was smaller than 3.

The NCuts algorithm was applied directly to the boundary images. The implementation of the NCuts used requires that the number of regions required be passed as a parameter. We used values of 5, 16 and 28, corresponding to the average number of regions obtained by respectively the WF QD level 2, humans and WF QD level 1. The average number of regions produced by each algorithm as well as the region number agreement are also shown in Table 1.

6.3 Discussion

The lowest GCE value in Table 1 (excepting humans) was obtained by the WF TD level 1. However, as the average number of regions for this method

[5] We used an implementation by J. Shi available here: `http://www.cis.upenn.edu/~jshi/software/`



(a) WF TD level 1 (b) WF QD level 1 (c) Human

(d) WF TD level 2 (e) WF QD level 2 (f) NCuts (16 regions)

Fig. 6. Histograms of the distribution of the mean GCE for each of the 100 test images for: (a, b, d, e) the four different applications of the Waterfall algorithm, (f) the NCuts algorithm with 16 regions, and (c) the human-human comparison (note that this has more than 100 values as the human segmentations for each image are compared using a leave-one-out approach).

is 51.4, it appears that the images are over-segmented. It is mentioned in [3] that as the GCE measure is tolerant of refinement (splitting of regions), an over-segmentation can result in a smaller GCE value. This method also has the smallest region number agreement.

The other three Waterfall-based methods produce GCE values between 0.21 and 0.23, even though the average number of segments is much higher for the WF QD level 1 than for the two level 2 results. The WF QD level 1 has the smallest GCE of three along with the highest region number agreement. The GCE distributions for these three methods shown in Figure 6(b), (d) and (e) are similar. Figure 7(a) shows the mean and standard deviations of the GCE obtained for each of the 100 images when comparing the segmentation obtained by the WF QD level 1 to the corresponding human segmentations. The large differences in the mean GCE as a function of the image, as well as the large standard deviations due to significant differences in the human segmentations are clearly visible.

Concerning the number of segments produced, level 1 of the Waterfall-based methods tends to be an over-segmentation of the image, whereas level 2 tends to be an under-segmentation. This can be seen when comparing the average

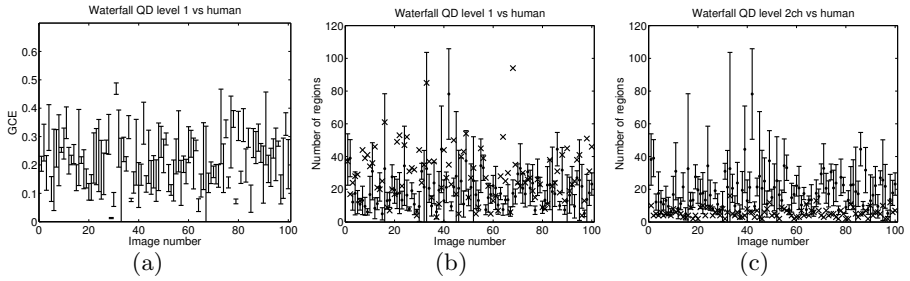


Fig. 7. (a) Mean and standard deviation of the GCE calculated for the WF QD level 1 with respect to the corresponding human segmentations for each test image. (b, c) Mean and standard deviation of the number of regions in the human segmentations for each of the test images (bars) and the number of regions in the (b) WF QD level 1 and (c) WF QD level 2 (crosses).

number of regions obtained (given in Table 1) with the average number of 16.8 obtained for the human segmentations. Figure 7(b) and (c) give a more detailed view of the number of regions obtained per image. The bars show the mean and standard deviations of the number of regions in the human segmentations for each image, while the crosses show the number of regions obtained respectively by the WF QD level 1 and level 2. The large variation in the number of regions in the human segmentations of some of the images are visible. Furthermore, one can see that the majority of crosses are above the error bars for level 1 and below for level 2. This suggests the introduction of an alternative (less strict) merging rule in the Waterfall algorithm.

The Waterfall-based approaches produce smaller global GCE values than the NCuts with 5 and 16 regions. The GCE for level 2 of the Waterfall methods, even with the small number of regions, is significantly lower than the GCE of the NCuts for 5 regions. This suggests that the regions found by the Waterfall method are a better match to the human segmentations. The NCuts with 16 regions has a GCE value similar to those of the majority of Waterfall methods. The distribution of these GCE values are shown in Figure 6(f). This method also has the highest region number agreement. The second smallest GCE value in Table 1 corresponds to the NCuts with 28 regions, nevertheless it is possible that this is again due to over-segmentation. The Waterfall-based approaches have the advantage that the number of regions do not need to be specified in advance. There is a version of the NCuts which determines the number of regions automatically [12], but we currently have no implementation of it.

7 Conclusion

We have compared a morphological Waterfall-based segmentation algorithm to the Normalised Cuts algorithm using the Berkeley Segmentation Benchmark. Both segmentation algorithms are applied to boundary images obtained from a

learning-based algorithm. These boundary images are not suitable for use with Watershed-based algorithms due to gaps in the boundary lines, a problem we have solved by calculating a distance function of the boundary images. Two types of distance function were tested, with one of them requiring no parameters as it operates directly on the greyscale images.

Based on the results of the benchmark, it is difficult to make a final pronunciation on which of the tested algorithms are better. For a small number of regions, the Waterfall algorithm has a lower GCE than the NCuts, but the GCE values are similar for segmentations with a higher number of regions. The Waterfall algorithm tends to produce too many regions at the first level of its hierarchy and too few at the second level. It should be possible to change the region merging criteria to improve this. It would also be interesting to test the version of the NCuts which does not require the number of regions to be specified in advance.

References

1. 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
2. Chen, Y., Wang, J.Z.: Image categorization by learning and reasoning with regions. Journal of Machine Learning Research **5** (2004) 913–939
3. 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. 8th Int'l Conf. Computer Vision. Volume 2. (2001) 416–423
4. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. on Pattern Analysis and Machine Intelligence **22** (2000) 888–905
5. Beucher, S., Meyer, F.: The morphological approach to segmentation: the watershed transformation. In Dougherty, E., ed.: Mathematical Morphology in Image Processing. Marcel Dekker (1993) 433–481
6. Martin, D., Fowlkes, C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE Transactions on Pattern Analysis and Machine Intelligence **26** (2004) 530–549
7. Lantuejoul, C., Beucher, S.: On the use of geodesic metric in image analysis. Journal of Microscopy **121** (1981) 39–49
8. Soille, P.: Morphological Image Analysis. second edn. Springer (2002)
9. Beucher, S.: Numerical residues. In: Mathematical Morphology and its Applications to Image Processing, Proc. ISMM'05. (2005) 23–32
10. Beucher, S.: Watershed, hierarchical segmentation and waterfall algorithm. In: Mathematical Morphology and its Applications to Image Processing, Proc. ISMM'94. (1994) 69–76
11. Marcotegui, B., Beucher, S.: Fast implementation of waterfall based on graphs. In: Mathematical Morphology and its Applications to Image Processing, Proc. ISMM'05. (2005) 177–186
12. Malik, J., Belongie, S., Leung, T., Shi, J.: Contour and texture analysis for image segmentation. International Journal of Computer Vision **43** (2001) 7–27