Image Segmentation Based on Height Maps

Gabriele Peters[1] and Jochen Kerdels[2]

[1] University of Dortmund
Department of Computer Science - Computer Graphics
Otto-Hahn-Str. 16, D-44221 Dortmund, Germany
[2] DFKI - German Research Center for Artificial Intelligence
Robotics Lab
Robert Hooke Str. 5, D-28359 Bremen, Germany

Abstract. In this paper we introduce a new method for image segmentation. It is based on a height map generated from the input image. The height map characterizes the image content in such a way that the application of the watershed concept provides a proper segmentation of the image. The height map enables the watershed method to provide better segmentation results on difficult images, e.g., images of natural objects, than without the intermediate height map generation. Markers used for the watershed concept are generated automatically from the input data holding the advantage of a more autonomous segmentation. In addition, we introduce a new edge detector which has some advantages over the Canny edge detector. We demonstrate our methods by means of a number of segmentation examples.

Keywords: segmentation, edge detection, watershed, height maps.

1 Introduction

In this paper we introduce a new method for image segmentation. It is based on a height map which is generated from the gray values of the input image. Among the different methods for image segmentation morphological watersheds have some advantages [1]. They yield more stable results in comparison to other segmentation concepts such as detection of discontinuities, thresholding, or region processing. But they also have a drawback. Watersheds work on height level images. The association with height maps refers directly to the input image. If an image is interpretable as topographic image, such as images of cells under a microscope, watersheds perform well. To apply watershed segmentation to arbitrary images such as photographs of natural objects we propose to generate a height map which characterizes the content of the image in an appropriate way. A simple interpretation of an arbitrary image, e.g., of a tree, as height map would make no sense. We introduce the derivation of an appropriate height map from an edge filtered version of the input image. For that purpose we propose a new edge detector related to the Canny edge detector, but endowed with some advantages. This enables us to apply the watershed concept for the segmentation of arbitrary images and to exploit its friendly properties also for difficult images such as those

W.G. Kropatsch, M. Kampel, and A. Hanbury (Eds.): CAIP 2007, LNCS 4673, pp. 612–619, 2007.

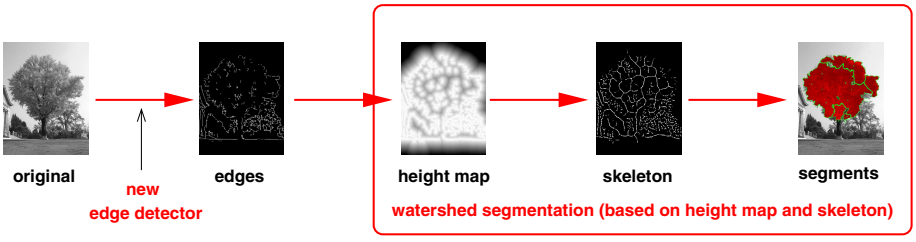

Fig. 1. Image Segmentation Based on Height Maps, Overwiew. The contributions of this paper are highlighted in red.

depicting natural objects. In addition, and in contrast to the general watershed concept, where *markers* are adopted to incorporate knowledge-based constraints in the segmentation process, we automatically generate markers from the height map and thus contribute to a more autonomous segmentation process. The segmentation based on height maps is derived in section 2, section 3 describes some results, and in section 4 we conclude the paper with a summary.

2 Image Segmentation Based on Height Maps

In fig. 1 an overview of the proposed image segmentation method is given. We start with the original image to be segmented. An edge detector is applied to the gray values of this image resulting in an edge image (subsection 2.2). From this binary edge image we generate a topographic image, the gray values of which can be interpreted as different height levels (subsection 2.3). Given this height map a skeleton image is derived, which is again a binary image (subsection 2.4). The height map is the source image for the watershed algorithm, which is applied utilizing the skeleton image as markers. This results in the final segmentation of the original image (subsection 2.5). The details of our approach, especially in a formalized version, are described in [2]. Here we restrict to the introduction of the basic principles of the proposed segmentation method. But first we start with a short retrospection of the concept of morphological watersheds (subsection 2.1).

2.1 Segmentation by Watersheds Revisited

Segmentation by morphilogical watersheds [3,4] requests a gray value image as input and is based upon the interpretation of this image as a topographic image, where different gray values code for different heights in a third dimension. Segmentation is accomplished by a slow flooding of this imaginary landscape. Water appears first in the deepest valleys and reaches higher landmarks at a later point in time. Any time the water impends to slop over a mountain crest a dam is erected to prevent the water from overflowing. The algorithm terminates when the whole landscape is flooded and only dams are visible. These dams constitute the segmentation of the image. The fact that each segment has its origin in a

local minimum of the height map can lead to an oversegmentation. A number of methods have been proposed dealing with this problem [5,6]. A suitable means to prevent oversegmentation is the utilization of so-called *markers*. They designate landmarks where water can emerge and thus, they restrict the number of segments in advance. Valleys, i.e., local minima, which are not labeled by a marker are not protected by dams. They are flooded eventually from one of the neighboring valleys and thus are assigned to the corresponding segment. It depends on the application at hand, where to place the markers. Originally markers have been regarded as an opportunity to incorporate knowledge-based constraints in the segmentation process [1]. Contrary to this opinion, we recognize the fact that an automatic generation of markers - as employed by our approach (and described in subsection 2.4) - holds the merit of a more automatic, independent, and self-contained segmentation process.

2.2 Edges

In this subsection we describe the derivation of a binary edge image from the input gray value image to be segmented. The edge image is the basis from which we derive the height map of the original image, which is needed as input for the watershed algorithm. As existing edge detectors display a series of disadvantages we develop a new edge detector, which is similar to the Canny edge detector [7] but has some advantages over it.

Proposed Edge Detector. The proposed edge detection proceeds in 7 steps:

1. *Simple Operator, Parameter d:* We employ a simple edge operator, which is given in the left diagram of fig. 4 for one direction (horizontal only) and one intensity transition (dark to light only) as an example. The empty entries contain zeros. Parameter d determines the distance between 1 and -1 in the mask of the operator, and thus the size of the utilized local window. A larger distance effects that *salient* edges in the image generate d-wide edges in the response of the operator. On the other hand, edges, that are caused by noise oder disturbed structures in the image, i.e., non-salient edges, result in thinner edges in the filter response.
2. *Gaussian Blur, Parameter b:* The so created preliminary edge image is filtered with a Gaussian blur the magnitude of which is regulated by the parameter b. Whereas wide edges are diminished by the blur filter only at their ends, thin edges are degraded as a whole.
3. *Potentiation, Parameter γ:* Now the lowpass filtered, still preliminary edge image is exponentiated by the exponent γ. As the domain of the edge image is $[0, 1]$, γ determines to which extent the values of the egde image are forced against zero.
4. *Thresholding, Parameter ϵ:* Afterwards we eliminate values below a threshold ϵ and set values above or equal the threshold to 1. This and the preceding steps result in a binary edge image with relatively wide edges.

5. *Thinning:* These wide edges are thinned to a width of one pixel only.
6. *Deletion, Parameter λ:* In addition, those edges are deleted the length of which is shorter than the average length of all edges muliplied by the factor λ.

An exemplification of these six steps is given in fig. 2.

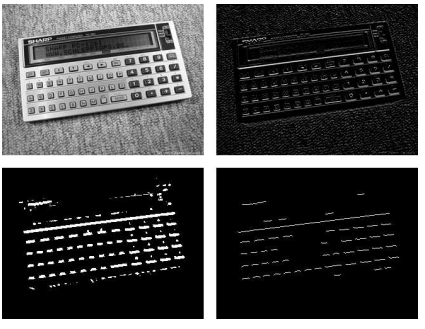

Fig. 2. Steps of the Edge Detector. The steps are illustrated for horizontal dark-to-light transitions only. Upper left: original image. Upper right: result after the first step of the algorithm. Lower left: result after carrying out steps 2 to 4. Lower right: resulting edge image after steps 5 and 6. The parameters used are $\{d = 2,\ b = 2,\ \gamma = 3.0,\ \epsilon = 0.01,\ \lambda = 1.0\}$.

7. *Joining Directions and Transitions:* In the final step the results from the edge detectors for the single directions and transitions are joined into the final edge image. Fig. 3 shows the complete edge image of the last example in the right image.
   It also visualizes the effects of different sets of parameters.

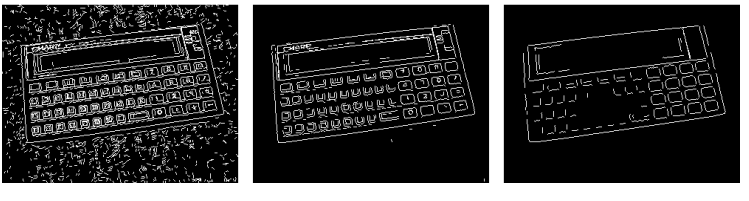

Fig. 3. Edge Images for Different Sets of Parameters. Left: $\{d = 1, b = 1, \gamma = 2.0, \epsilon = 0.01, \lambda = 1.0\}$. Middle: $\{d = 1, b = 1, \gamma = 2.5, \epsilon = 0.01, \lambda = 1.0\}$. Right: the same parameters as in fig. 2, i.e., $\{d = 2, b = 2, \gamma = 3.0, \epsilon = 0.01, \lambda = 1.0\}$.

Advantages in Comparison with the Canny Edge Detector. Most of the steps are also carried out by the Canny edge detector, though in different order. Without going into detail, the principle steps of the Canny operator are Gaussian filtering of the image, edge detection, and thresholding. In particular, one difference of the proposed edge detector to the Canny detector is the potentiation of the blurred edge image in step 3. In combination with the following step (thresholding) this introduces an "exponential thresholding" in contrast to

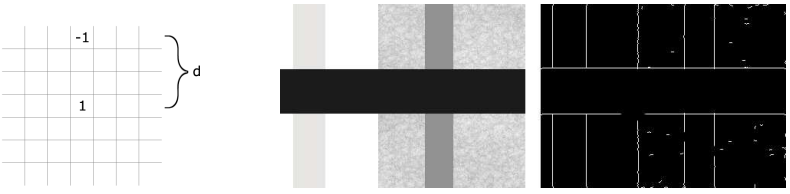

Fig. 4. Simple Edge Operator and Advantage of the Proposed Edge Detecor. Left: simple edge operator. Middle: original image. Right: resulting edge image. Salient edges in the original image are preserved even if they are of low contrast such as the edges of the light, vertical bar on the left. On the other hand, edges of similar contrast are largely suppressed if they are not salient such as the edges in the texture on the right.

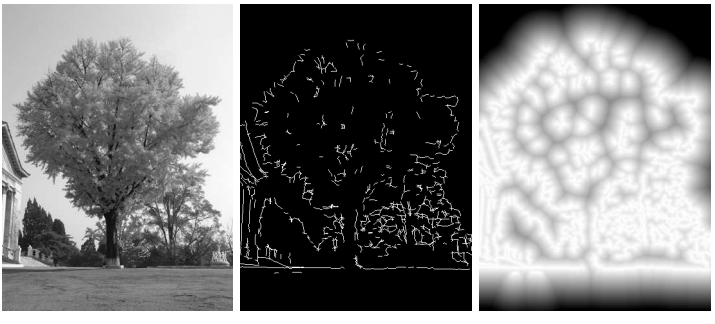

Fig. 5. Edge Image and Height Map. Left: original gray value image. Middle: binary edge image. Right: gray value height map. Note the gap in the top of the tree outline in the edge image. The height map compensates perfectly for this deficiency.

the linear thresholding of the Canny detector. This kind of thresholding seems to be easier tunable. Experiments reported in [2] suggest a more robust behavior of the proposed edge detector. The same threshold value provides good results for a larger field of application than the Canny edge detector. Another advantage is illustrated in fig. 4. In step 1 edges of different strength are generated, depending on the saliency (not the contrast) of the edges in the original image: the more salient an edge in the source image, the wider the corresponding edge in the (intermediate) edge image. The subsequent application of the blur filter diminishes the thin edges (derived from non-salient edges in the original image) and thus boosts the wide, i.e., salient edges. In addition, the results of our edge detector seem to have similar advantages as those of Canny operators augmented with surround suppression such as described in [8], though it does not inhibit its surroundings actively. This can be seen in the edge image of fig. 7.

2.3 Height Map

As already mentioned we want to apply a watershed algorithm for segmentation, thus we need a height map. In this subsection we derive from the binary edge

image a gray value topographic image, which characterizes the original image in an adequate way. First we define a maximal distance d_{max}. It determines the maximal distance of a point in the edge image to the closest edge. Points with a larger distance from an edge are defined to be on the same height. That means that only gaps of a maximal size of $2d_{max}$ are tolerable in the edge image. For the images used, with a resolution of 640×480, $d_{max} = 32$ to 64 pixels provides good results. The idea is now that for each point in the edge image the minimal distance to the closest edge is determined. For that purpose we start a region growing from each point e of the edge image marked as an edge. For each image point p under consideration during this region growing we first verify whether a distance value has already been assigend to it and whether this distance value is smaller than the current distance d_{cur} to the origin e of the region growing. If not, $d_{max} - d_{cur}$ is assigned to p. In addition, the neighboring points of p are stored as future candidates in the region growing. The region growing stops if either no more candidates for growing exist or the maximal distance d_{max} is reached. The right image of fig. 5 shows an example of a height map generated according to the preceding specification. Another example is shown in fig. 7.

2.4 Skeleton

As already pointed out in subsection 2.1 markers are used in combination with watershed algorithms to prevent the result from beeing oversegmented. The binary skeleton image, which we derive in this subsection, will be used as marker for the watershed algorithm we apply to the height map. By the application of Laplace operators to the height map one obtains a kind of gray value skeleton image. The coarseness of these skeletons can be controlled by the size of the Laplace operator as illustrated in fig. 6. For our examples we used a 3×3 filter. After the application of the Laplace filter we binarize the result with a threshold and, in addition, we delete those of the remaining edges that are thinner than

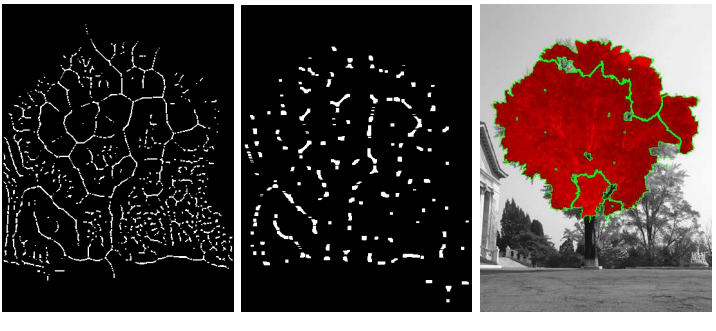

Fig. 6. Skeleton Images and Final Segmentation. Left: skeleton image of the same original image as depicted in fig 5, generated with a 3×3 Laplace filter. Middle: skeleton image generated with an 11×11 Laplace filter. Right: parts of the segmentation after applying the watershed algorithm with the left skeleton image as marker.

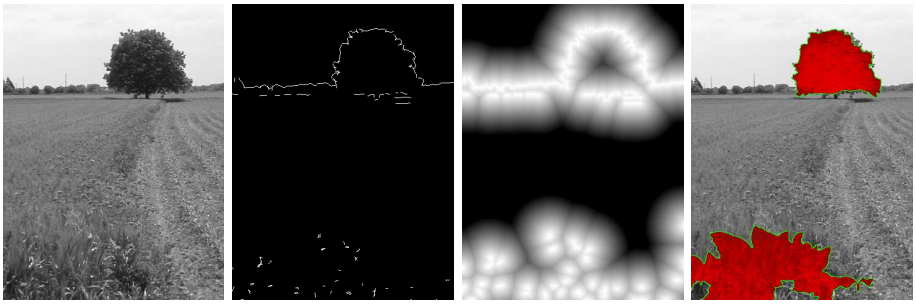

Fig. 7. Example of Image Segmentation Based on Height Maps. First: original image. Second: edge image. Third: height map. Fourth: parts of the resulting segmentation.

half of the filter size. Every white pixel of the resulting binary skeleton image acts as a marker for the following application of the watershed algorithm.

2.5 Segmentation

In this final step we apply the watershed algorithm as described in subsection 2.1 to the height map derived in subsection 2.3 utilizing the skeleton image as markers. As even the usage of markers cannot always prevent the result from beeing oversegmented similar neighboring segments are fused in a final postprocessing step. As a measure of similarity between two segments three values are compared: the average gray value, the variance of the gray values, and the entropy (calculated from the histogram of the gray values). Average, variance, and entropy of neighboring segments are compared, and only if they resemble significantly both segments are fused. As threshold for the difference between the average we used 0.1, for the variance 0.05, and for the entropy 0.5. The right image of fig. 6 shows the interesting segments of the result after the last fusion step has been carried out.

3 Results

For the evaluation of the proposed segmentation method we chose images depicting natural objects such as trees as these are typically difficult to segment. A first example is depicted in the figures 5 and 6. The height map compensates for gaps in the outline of the tree in the edge image. The resulting segmentation of the tree is fairly well accomplished and the foreground tree is separated from the tree in the background although these image areas are quite similar in terms of texture. Regrettably, the final fusion of similar segments turned out to be the weak point of the whole segmentation process as it was difficult to choose the values of the thresholds for average, variance, and entropy in such a way that neither an oversegmentation nor an undersegmentation occured, and this still for a series of different examples. But in the second example displayed in fig. 7 our approach provided one segment for the tree and another for the dominant tuft

in the foreground which corresponds quite well to our human perception. The separation between the tuft and the surrounding image areas can be regarded as difficult, because they are quite similar in structure. In addition, here again a gap in the contour of the tree poses no challenge for the further processing.

More examples are discussed in [2].

4 Summary

We proposed a method to derive a height map from an arbitrary gray value image, which characterizes its content in such a way that the application of the watershed concept to the height map provides a proper segmentation of the image. This enables the application of the watershed segmentation also to images that are not similar to a topographic image. That means the watershed algorithm can provide now better segmentation results on difficult images, e.g., images of natural objects, than without the intermediate height map generation. The markers, usually introduced to the watershed algorithm to incorporate knowledge-based constraints, are generated automatically from the height map. This holds the merit of a more automatic, independent, and self-contained segmentation process. In addition, we introduced a new edge detector. Its advantages in comparison with the Canny edge detector are twofold. By the application of the blur filter to an intermediate edge image instead of the original image (as the Canny detector does) we derive an edge image which better distinguishes between salient and non-salient edges. The second merit of the proposed edge detector is an easier to handle thresholding which seems to provide more robust results.

Acknowledgments. This research was funded by the German Research Association (DFG) under Grant PE 887/3-1.

References

1. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Prentice-Hall, Englewood Cliffs (2002)
2. Kerdels, J.: Dynamisches Lernen von Nachbarschaften zwischen Merkmalsgruppen zum Zwecke der Objekterkennung. Diploma Thesis, University Dortmund (2006)
3. Serra, J.: Image Analysis and Mathematical Morphology. Ac. Press, NY (1988)
4. Beucher, S., Meyer, F.: The Morphological Approach of Segmentation: The Watershed Transformation. Mathematical Morphology in Image Processing. Marcel Dekker, New York (1992)
5. Najman, L., Schmitt, M.: Geodesic Saliency of Watershed Contours and Hierarchical Segmentation. IEEE PAMI 18(12), 1163–1173 (1996)
6. Bleau, A., Leon, J.: Watershed-Based Segmentation and Region Merging. Computer Vision and Image Understanding 77(3), 317–370 (2000)
7. Canny, J.: A Computational Approach to Edge Detection. IEEE PAMI 8(6) (1986)
8. Grigorescu, C., Petkov, N., Westenberg, M.A.: Contour and Boundary Detection Improved by Surround Suppression of Texture Edges. Image and Vision Computing 22(8), 609–622 (2004)