

Watersheds and Normalized Cuts as basic tools for Perceptual Grouping

Johan De Bock, Patrick De Smet and Wilfried Philips
Dep. Telecommunications and
Information Processing (TELIN/TW07)
Sint-Pietersnieuwstraat 41, B-9000 Gent, Belgium
Phone: +32 9 264.79.66, Fax: +32 9 264.42.95
E-mail: jdebock@telin.UGent.be

Abstract—In this paper we present a new image segmentation framework based on watersheds and normalized cuts. We propose a new way to use the normalized cut graph partitioning algorithm for image segmentation. We show that it is possible to apply the normalized cut algorithm, originally applied to pixels, to micro segments. This severely reduces the computational demand of the normalized cut algorithm. A floating point based rainfalling watershed algorithm will create the initial micro segmentation. We investigate the noise robustness of the complete segmentation algorithm and show the results we obtained on a photographic image.

Keywords—image segmentation, rainfalling watershed, graph partitioning, normalized cut

I. INTRODUCTION

Image segmentation still remains a fundamental task in computer vision. It is crucial in a number of applications, ranging from image coding and tracking to content-based image retrieval and object recognition. A correct segmentation is important in the sense that the quality of the segmentation strongly conditions the following analysis.

Graph partitioning is a fairly new and promising approach to perceptual grouping or image segmentation. It treats image segmentation as a graph partitioning problem. The basic scheme is to construct a graph with edge weights from the information contained in the individual pixels of a digital image. That graph is then partitioned following a global criterion. It thus partitions or segments an image from a global point of view.

A global criterion that has shown much promise is the normalized cut criterion [1] that solves the main weakness of the traditional minimum cut [2], namely the tendency to cut off very small segments. The core computational technique of the normalized cut algorithm is a generalized eigenvalue problem. Although it is an elegant way to optimize the normalized cut criterion, the computational complexity of an eigenvalue decomposition is very high.

The question then remains how we can reduce the

computational demand without altering the technique too much. In this paper we try to reduce the computational demand by reducing the size of the graph. In the original description of the normalized cut algorithm for image segmentation, one node corresponds with one pixel, so the number of nodes in the graph equals the number of pixels in the image.

In this paper we replace the individual pixels by micro segments in order to reduce the number of nodes in the graph. It is very important that the micro segmentation will already yield a meaningful segmentation, i.e. the micro segments must be homogeneous and the edges contained in the image must correspond to segment boundaries.

Watershed segmentation is a technique that delivers these requirements. In this paper we use a floating point based rainfalling watershed segmentation [3] that results in a segment label image and thus clearly defined segments and segment boundaries.

The paper is organized as follows: in section II we describe the floating point based rainfalling watershed segmentation. In section III we explain the original normalized cut algorithm for image segmentation and the changes that are needed when we apply it to micro segments. In section IV we display the results we obtained with our complete segmentation framework on a real world image and illustrate the noise robustness on an artificial image. We finally draw some conclusions in section V.

II. WATERSHED SEGMENTATION

To obtain the micro segments we use a watershed transform. This transform considers its input as a topographic landscape in which “valleys” should correspond to the interior pixels of segments, whereas the “mountains” should correspond to the pixels in the neighborhood of the boundaries of segments. This correspondence is delivered by using the gradient magnitude of the input image as a topographic landscape for the watershed transform. The watershed transform extracts the “mountain rims” from the

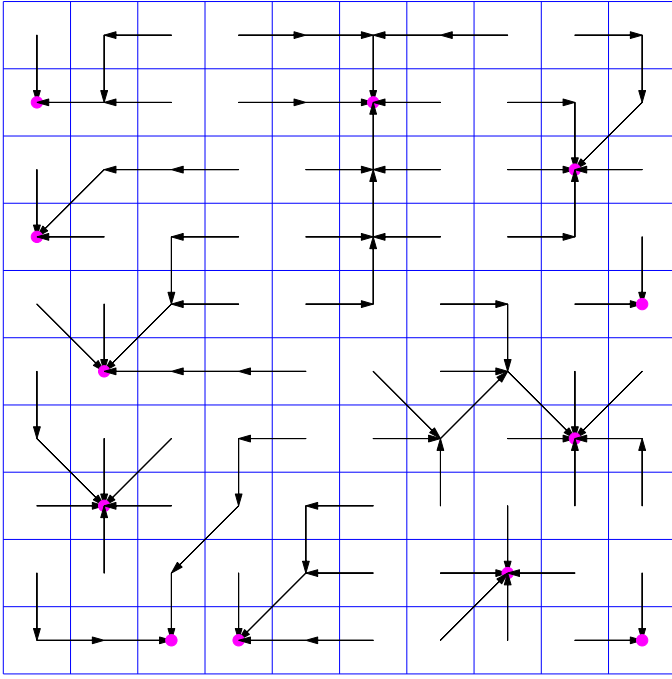


Fig. 1. Steepest descent directions

topographic landscape and those mountain rims then delineate the micro segments in the image.

As already mentioned, we use a floating point based rainfalling watershed transform to obtain the micro segments. Traditional watershed immersion implementations [4] require a quantized topographic landscape which must not contain too many levels for efficiency reasons. Our implementation uses a gradient magnitude image with floating point accuracy as input and consequently is not affected by loss of information due to quantization.

Conceptually, we track the path that a virtual droplet of water would follow when it falls on the topographic landscape at a certain pixel position and flows down the slopes of the topographic landscape (hence the name rainfalling algorithm). In the implementation this is done by calculating the steepest descent neighbor for each pixel. To be more precise, we search for the lowest pixel in an eight-neighborhood around a central pixel. That pixel should also be lower than the central pixel. We keep track of the steepest descent neighbors by first assigning a unique label to each pixel and then saving the label of the steepest descent neighbor in an array indexed by the labels of the central pixels. If there is no descent possible we save the label of the central pixel. A visualization of the steepest descent directions for an image of 10 by 10 pixels is given in figure 1. The pixels marked with a circle in the middle are pixels from where there is no descent possible. Hence, they are the local minima of the topographic landscape. Every group of pixels that is connected by the same tree of

arrows leading to a local minimum will now form one segment. We thus get one segment for every local minimum. The tracking of the pixels that belong to one tree is done by viewing the earlier defined array of labels as a label equivalence table and propagating the labels until each pixel has the label of one of the local minima. One thing now still remains to be done. It is possible that a local minimum does not consist of just one pixel, but that it consists of numerous (eight-neighborhood) connected pixels all at the same topographic height. An example of such a local minimum can be seen in figure 1; the local minimum at the bottom left consists of two pixels. Up until now those pixels still have different labels and thus would still belong to different segments. Logically they should belong to the same segment. This situation is handled by using a connected components algorithm to clearly determine the connected clusters of local minima. This algorithm assigns a different label to each separately connected group of local minima. Consequently, the connected local minima will share the same label. We then can use the new labels for the local minima to do the final relabeling. After copying the labels to a matrix (image) structure, we finally have a micro segmentation of the original image in the form of a segment label image; i.e. an image with for each pixel a label of the micro segment to which the pixel belongs.

III. NORMALIZED CUT APPLIED TO MICRO SEGMENTS

A. Original normalized cut algorithm for image segmentation

In this section we will give an explanation of the original normalized cut algorithm for image segmentation. As already mentioned the basic concept is to treat image segmentation as a graph partitioning problem. We thus need to build up a graph G with edge weights from the information contained in the individual pixels of the image. In the original normalized cut algorithm the authors [1] constructed the weighted graph by considering each pixel as a node. The edges are determined by the proximity of pixels: each pair of pixels (nodes) is connected by an edge if they are located within a distance r from each other. This procedure completely determines the connectivity of the graph. Now the edge weights need to be determined based on the similarity and distance between pixels. This is done by applying the following formula for each edge:

$$w_{ij} = e\left(-\frac{\|I_i - I_j\|^2}{\sigma_I^2}\right) \cdot e\left(-\frac{\|X_i - X_j\|^2}{\sigma_X^2}\right)$$

with w_{ij} the weight of the edge that connects node i with node j , I_i the intensity of pixel i and X_i the spatial location of pixel i . If there is no edge between node i and node j

then $w_{ij} = 0$. The effect of this procedure is that the pixels that are located very close to each other and that have a very small difference in intensity get a high weight on the edge that connects them. The pixels that are located very far from each other and that have a very big difference in intensity get a low weight on the edge that connects them. All those weights w_{ij} are saved in a matrix \mathbf{W} . This matrix will be the input of the core normalized cut algorithm.

Now the algorithm tries to partition the nodes V of the graph G in two disjoint sets A and B . The weight of the edges that stay within the same set should be as high as possible and the weight of the edges that connect the two sets should be as low as possible. This is the translation of the general view on segmentation: the internal pixels of segments should be homogeneous and there must be enough difference between neighboring segments. This concept is translated by the authors to the minimalization of the normalized cut criterion:

$$Ncut(A, B) = c(A, B) \left(\frac{1}{c(A, V)} + \frac{1}{c(B, V)} \right)$$

$$\text{with } c(X, Y) = \sum_{i \in X, j \in Y} w_{ij}$$

The weight of the edges that stay within the same set are contained in $c(A, V)$ and $c(B, V)$, the weight of the edges that connect the two sets are contained in $c(A, B)$.

Now an algorithm has to be created that can minimize the normalized cut criterion. Unfortunately, minimizing this criterion constitutes an NP-complete problem but the authors showed that when the normalized cut problem is embedded in the real value domain an approximate discrete solution can be found efficiently. By using matrices they reformulated $Ncut(A, B)$ to:

$$\frac{(\mathbf{1} + \mathbf{x})^T (\mathbf{D} - \mathbf{W})(\mathbf{1} + \mathbf{x})}{k \mathbf{1}^T \mathbf{D} \mathbf{1}} + \frac{(\mathbf{1} - \mathbf{x})^T (\mathbf{D} - \mathbf{W})(\mathbf{1} - \mathbf{x})}{(1 - k) \mathbf{1}^T \mathbf{D} \mathbf{1}}$$

with \mathbf{x} being an indicator vector, $x_i = 1$ if node i is in set A and $x_i = -1$ if node i is in set B , \mathbf{D} being a diagonal matrix with $d_i = \sum_j w_{ij}$ on its diagonal and

$$k = \frac{\sum_{x_i=1} d_i}{\sum_i d_i}.$$

After some transformations, defining $b = k/(1 - k)$ and setting $\mathbf{y} = (\mathbf{1} + \mathbf{x}) - b(\mathbf{1} - \mathbf{x})$, the normalized cut problem can be translated in the following formula:

$$\min_{\mathbf{x}} Ncut(\mathbf{x}) = \min_{\mathbf{y}} \frac{\mathbf{y}^T (\mathbf{D} - \mathbf{W}) \mathbf{y}}{\mathbf{y}^T \mathbf{D} \mathbf{y}}$$

with the conditions $y_i \in \{1, -b\}$ and $\mathbf{y}^T \mathbf{D} \mathbf{1} = 0$. This expression is known as the Rayleigh quotient, if \mathbf{y} is relaxed

to take on real values, it can be minimized by solving the generalized eigenvalue problem:

$$(\mathbf{D} - \mathbf{W}) \mathbf{y} = \lambda \mathbf{D} \mathbf{y}.$$

The eigenvector \mathbf{y}_1 corresponding with the second smallest eigenvalue is the real valued solution of the normalized cut problem. To finally get the approximate discrete solution the eigenvector is split into two parts by mapping the real values to the discrete set of values $\{1, -b\}$. This mapping is done by sorting the real values of \mathbf{y}_1 and evaluating $Ncut(A, B)$ for a few evenly spaced splitting points. The split point that produces the lowest $Ncut(A, B)$ will then finally give us the approximate solution: the disjoint sets A and B together with their $Ncut(A, B)$ value.

B. Adaptation of the normalized cut algorithm to a micro segment input

The part that needed to change to adapt the original normalized cut algorithm for image segmentation to a micro segment based input was the calculation of the graph G : the build up of the connectivity of the graph based on the proximity of the segments and the calculation of the weights on the edges based on the similarity and distance between segments, in other words the calculation of the matrix \mathbf{W} .

The topology of individual pixels is known beforehand, it is always a matrix of pixels and we can access it directly when we are building up the connectivity of the graph G based on pixels. To be more precise, for each pixel we know the pixels that are adjacent to this pixel and can access them directly. For segments this information is not directly accessible and it depends on the given micro segmentation. We have to extract this information from the segment label image given by the watershed segmentation. We accomplish this by creating a region adjacency graph [5] from the segment label image.

The region adjacency graph is a graph structure where each node represents a distinct segment of the image partition and where an edge connecting two nodes represents the fact that the two segments associated with those nodes are adjacent. We construct the region adjacency graph by doing a raster scan of the segment label image, for every two different neighboring (eight-neighborhood) labels i, j we encounter, we set $A_{ij} = 1$. After completing the scan the region adjacency graph is represented by the adjacency matrix \mathbf{A} with $A_{ij} = 1$ if segments i and j are adjacent and $A_{ij} = 0$ if they are not. Now we can directly access the topology of the segments.

The process of determining the edges in the graph G based on proximity now has to be adapted to segments.

For pixels this was done by connecting each pair of pixels (nodes) by an edge if they were located within a distance r from each other. The logical translation of this procedure to segments is connecting each pair of segments (nodes) by an edge if their centroids are located within a distance r from each other. The problem with this procedure is that the distances are not known beforehand and thus have to be calculated for each pair of segments. This is too cumbersome so we need another type of proximity that can be easily calculated. One that can easily be calculated is the proximity of two segments expressed by the minimum path length between the two nodes corresponding with the segments in the region adjacency graph \mathbf{A} . The simple calculation \mathbf{A}^k on the adjacency matrix gives us the graph we would get if we connect a pair of nodes by an edge if there is a path of at most length k between them in \mathbf{A} . We will call \mathbf{A}^k the higher order region adjacency graph; this graph is the graph G adapted to segments. We call k the adjacency order parameter.

We still need to define the calculation of the weights on the edges of G based on the similarity and distance between segments. Here we can use the logical translation of the approach taken for segments:

$$w_{ij} = e\left(-\frac{\|I_i - I_j\|^2}{\sigma_I^2}\right) \cdot e\left(-\frac{\|X_i - X_j\|^2}{\sigma_X^2}\right)$$

with w_{ij} the weight of the edge that connects node i with node j , I_i the mean intensity of segment i and X_i the spatial location of the centroid of segment i . Now we have completely determined \mathbf{W} based on a micro segmentation. From this point on we can use the normalized cut optimization technique that we described earlier to partition the micro segmentation.

Finally, we implemented the necessary data structures and procedures in order to apply the algorithm recursively. We thus are able to repartition the previously formed partitions using the same normalized cut optimization technique. We can regulate the recursion with two parameters. Parameter d sets a unconditional limit on the recursion depth, it does not consider any output of the normalized cut algorithm. Parameter s sets the maximum allowed $Ncut(A, B)$. We can consider $Ncut(A, B)$ as a measure of the optimality of the partition. By stopping the partitioning in sets A and B if $Ncut(A, B) > s$ we prevent the creation of bad partitions.

IV. RESULTS

A. Noise robustness

We first examine the performance of the normalized cut algorithm applied to a micro segmentation with respect to



Fig. 2. Artificial test image

noise. Such a performance evaluation can be realized by studying the behavior of the complete algorithm on an artificial image. We generated the artificial image displayed in figure 2. It is an image of 256 by 256 pixels containing two regions of different but constant intensity. The domain of the intensity values is defined as $[0, 1]$. The intensity of the foreground region is set to $2/3$ and the intensity of the background is set to $1/3$. Now we add uncorrelated noise of a Gaussian distribution with mean 0 and variance σ_n^2 to the artificial image and let the algorithm do one partition iteration on the resulting image with the following parameters $\sigma_I = 0.047, \sigma_X = 33, k = 6, d = 1$ for the normalized cut algorithm. For the watershed segmentation no parameters have to be set.

We gradually increase the variance σ_n^2 of the noise in steps of 0.001 and visually inspect the quality of the segmentation. For variance $\sigma_n^2 = 0.001$ the watershed algorithm gives the micro segmentation shown in figure 3. After applying the normalized cut algorithm on the graph G constructed from the micro segmentation, a correct segmentation is produced, it is shown in figure 4.

The segmentation algorithm continued to give a correct segmentation of the artificial image with added noise until we tried it on the image corrupted by noise with variance $\sigma_n^2 = 0.005$. The resulting segmentation is shown in figure 5. We can see a little red segment that disturbs the otherwise perfect segmentation. This effect increases if we look at the results on the next noise levels. In figure 6 the result is given for $\sigma_n^2 = 0.035$. The algorithm performed

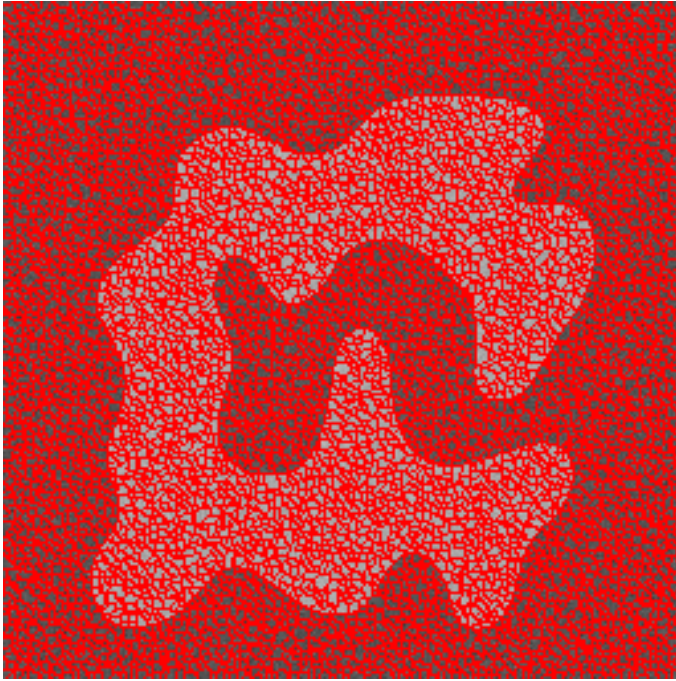


Fig. 3. Micro segmentation, $\sigma_n^2 = 0.001$

only one partitioning step, so naturally you would expect a partition with two segments corresponding with the sets A and B .

The creation of those little segments can be explained by looking at the eigenvector image: the two dimensional visualization of the eigenvector \mathbf{y}_1 , created by rescaling the real values of \mathbf{y}_1 and mapping them on the corresponding segment in the micro segmentation. The eigenvector image for $\sigma_n^2 = 0.035$ is displayed in figure 7. If we do the normal one dimensional mapping of the real values of \mathbf{y}_1 onto the discrete set $\{1, -b\}$ to determine the sets A and B , the white spots in the black area will always be in the same set as the white area and vice versa. Segments belonging to set A can be spatially completely surrounded by segments belonging to set B and vice versa. This is the reason why it is possible that one partitioning step can create many (unwanted) segments.

We solved this problem by implementing an extra step that also uses spatial (two dimensional) information. We first limit the region adjacency graph \mathbf{A} by discarding the nodes of set B and keeping the nodes of set A . Then we search for the largest connected subgraph in that graph. This will give us a new set of nodes A and consequently $B = V \setminus A$. We repeat this procedure for the new sets but now we keep the nodes of set B and construct a new set B . This step will filter out the small isolated segments. The final result is shown in figure 8.

The algorithm also produced a good result for the noise

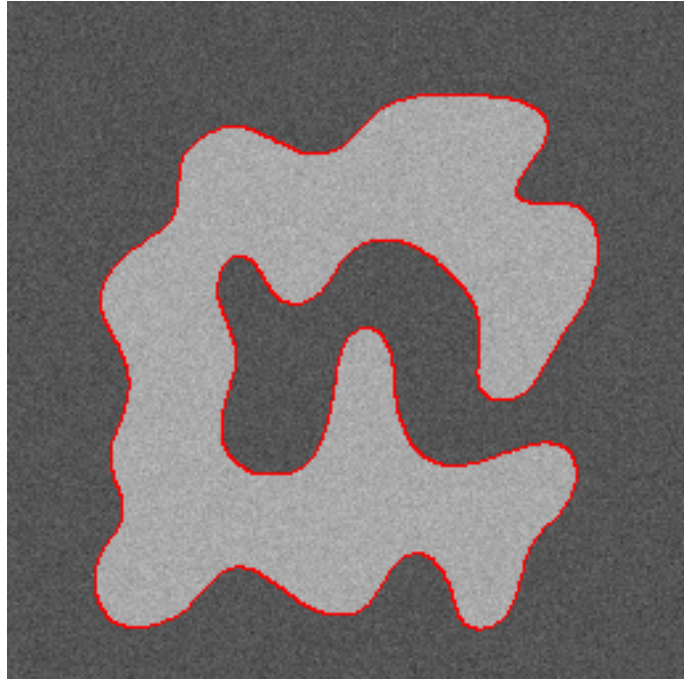


Fig. 4. Segmentation result, $\sigma_n^2 = 0.001$

levels $\sigma_n^2 = 0.036, 0.037, 0.038$. But for those levels only one small segment was formed in the first normalized cut partitioning step. After doing one extra partitioning step ($d = 2$), the correct segmentation was also found. This is displayed in figure 9 for noise variance $\sigma_n^2 = 0.038$.

To conclude the noise robustness evaluation, we show the segmentation result on the image corrupted with noise of variance $\sigma_n^2 = 0.039$ in figure 10. The algorithm has severe difficulties to find the correct segmentation.

B. Segmentation of a photographic image

We now display the results we obtained with our complete segmentation framework on a real world image. We used the image displayed in figure 11 which is part of the Berkeley Segmentation Dataset [6]. It is an image with dimensions 321 by 481 and thus in total 154401 pixels. The micro segmentation produced by the watershed algorithm consists of 12983 segments. We thus have achieved a severe reduction of the number of nodes in the graph G . We applied the normalized cut algorithm recursively and unsupervised with the following parameters: $\sigma_I = 0.022, \sigma_X = 3, k = 2, s = 0.001$. This produces the segmentation given in figure 12. This figure shows an acceptable segmentation; only a small part of the border of the vase could not be found. This can be explained by the blending of the vase with the background in that part of the image.

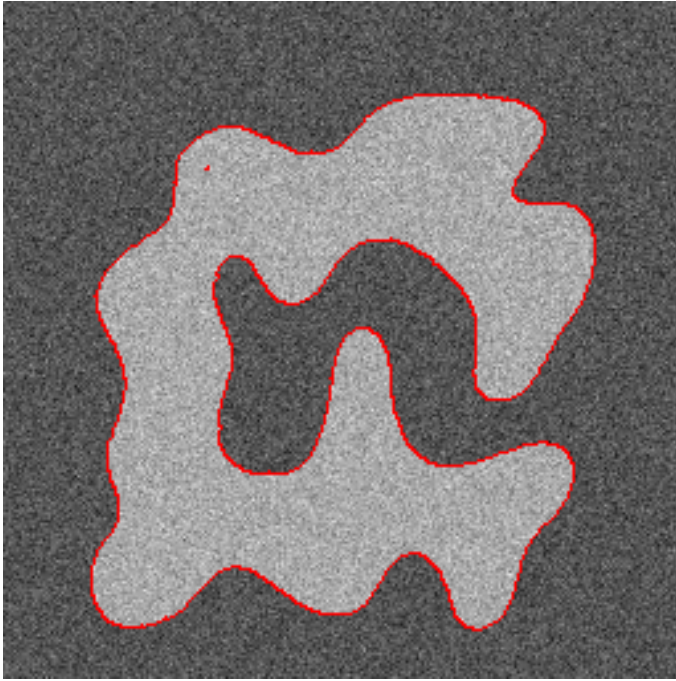


Fig. 5. Segmentation result, $\sigma_n^2 = 0.005$

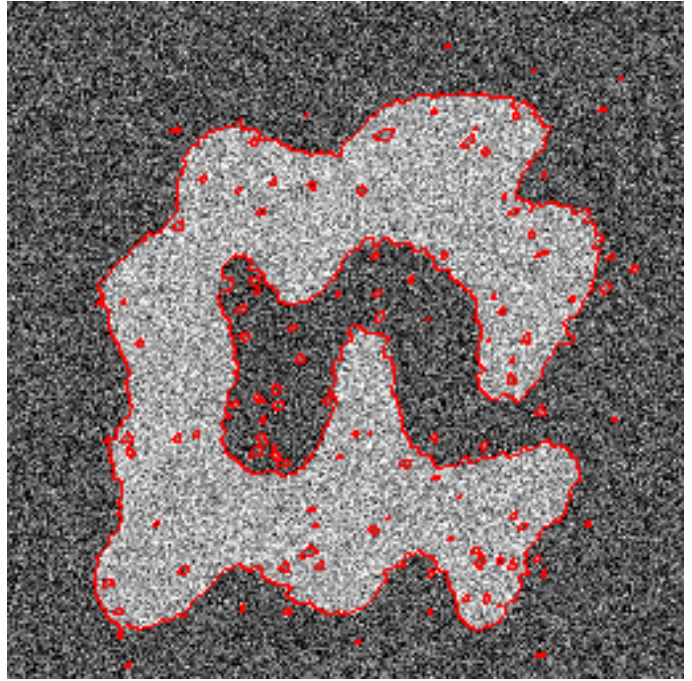


Fig. 6. Segmentation result, $\sigma_n^2 = 0.035$

V. CONCLUSION

We first described a floating point based rainfaling watershed algorithm that yields a meaningful micro segmentation of an input image. We then applied the normalized cut graph partitioning algorithm to a higher order region adjacency graph. We demonstrated how we could construct this graph from the information contained in the micro segmentation. By using micro segments instead of pixels, we severely reduced the number of nodes and consequently the computational demand of the normalized cut optimization technique. We investigated the noise robustness of the complete segmentation framework by segmenting an artificial test image with added noise. Up until noise level $\sigma_n^2 = 0.038$ the algorithm did not have any problems to find a good segmentation. We finally showed that the algorithm produced a visually very appropriate segmentation of a photographic image.

REFERENCES

- [1] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.
- [2] Z. Wu and R. Leahy, "An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1101–1113, 1993.
- [3] P. De Smet, "Segmentation and analysis of digital video sequences," Ph.D. dissertation, University Ghent, 2002.
- [4] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE Transactions on*

Pattern Analysis and Machine Intelligence, vol. 13, no. 6, pp. 583–598, 1991.

- [5] T. Pavlidis, *Structural Pattern Recognition*. Springer, 1977.
- [6] "The berkeley segmentation dataset and benchmark." [Online]. Available: <http://www.cs.berkeley.edu/projects/vision/grouping/segbench/>



Fig. 7. Eigenvector image, $\sigma_n^2 = 0.035$

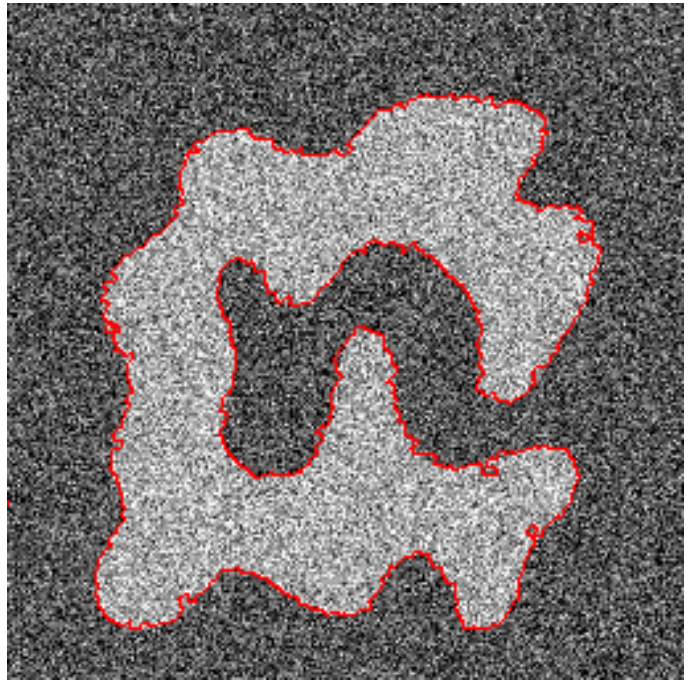


Fig. 9. Filtered segmentation result, $\sigma_n^2 = 0.038$

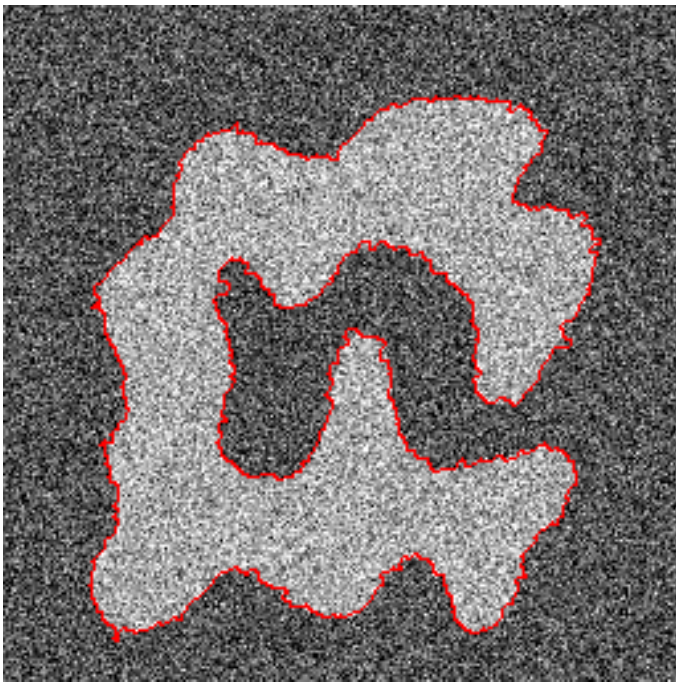


Fig. 8. Filtered segmentation result, $\sigma_n^2 = 0.035$

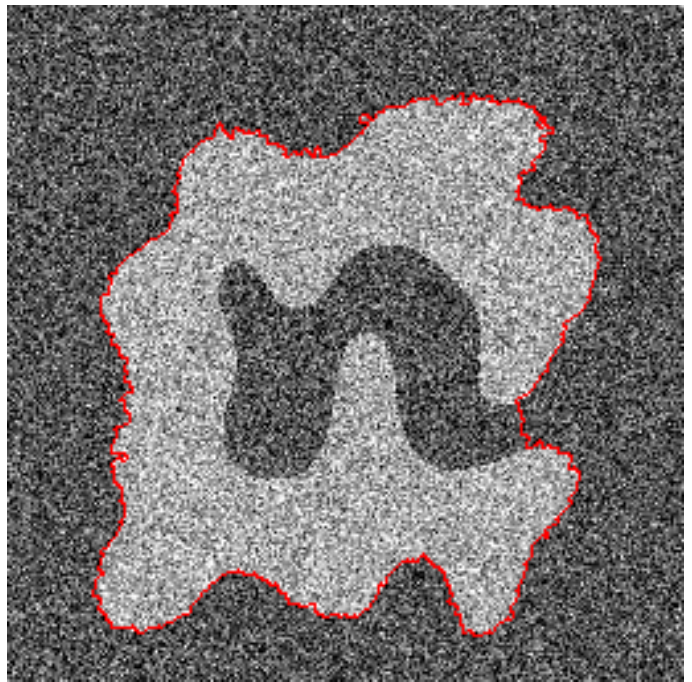


Fig. 10. Filtered segmentation result, $\sigma_n^2 = 0.039$



Fig. 11. Photographic test image



Fig. 12. Segmentation of the photographic test image