Local Watershed Operators for Image Segmentation

Hüseyin Tek and Hüseyin Can Aras

Imaging and Visualization, Siemens Corporate Research, Inc.
755 College Road East, Princeton NJ 08540, USA

Abstract. In this paper, we propose local watershed operators for the segmentation of medical structures. Watershed transform is a powerful technique to partition an image into many regions while retaining edge information very well. Most watershed algorithms have been designed to operate on the whole or cropped image, making them very slow for large data sets. In this paper, we propose a computationally efficient local implementation of watersheds. In addition, we show that this local computation of watershed regions can be used as an operator in other segmentation techniques such as seeded region growing, region competition or markers-based watershed segmentation. We illustrate the efficiency and accuracy of the proposed technique on several MRA and CTA data.

1 Introduction

Recent technological advances in imaging acquisition devices increase the spatial resolution of image data significantly. For example, new multi-detector CT machines can produce images with sizes as large as 512x512x1000. Thus, segmentation algorithms for these data sets need to operate locally in order to be computationally efficient since there is limited amount of time and memory available. While cropping data via user defined region of interest may be a solution for well localized pathologies, the user selected regions can still be very large in many applications, e.g. vascular segmentation or bone removal in CTA. Alternatively, images can be thresholded to reduce the area, where the segmentation and visualization algorithms need to operate, however, usually at the expense of removing anatomically important structures from the data.

We focus on the watershed-based segmentation of medical images in this paper. In medical image analysis, accurate detection of object boundaries are extremely important for quantification reasons, thus, making edge-based algorithms popular. While advances in edge detection algorithms increase the accuracy and performance of edge detectors, they are still not robust enough for many practical applications because edge grouping and linking (especially in 3D) is still quite a difficult problem. Unlike edge detectors, watershed transforms [8,5] produce closed contours and give good performance at junctions and places, where the object boundaries are diffused. Unfortunately, most watershed transforms are designed to operate on the whole image, which becomes a computational problem for large data sets.

In this paper, we propose a local implementation of watershed transform. Stoev and Strasser [7] also attempted to compute watersheds locally, which is discussed in Section 2. Our algorithm is based on filling operations from a user selected point. In the proposed algorithm, for each region to be filled accurately, its immediate two outer layers must be represented and filled at the same time. This *three layer* representation makes a user selected region (or catchment basin) [1] and its watershed lines to be computed correctly and locally. While the proposed algorithm can be used for the local computation of traditional watershed transform, it can also be used as an operator in segmentation algorithms. Specifically, instead of using pixel-based growth in many locally operating segmentation algorithms, regions can be used via the proposed local watershed operators. The incorporation of regions into the local segmentation algorithms allows much more accurate localization of object boundaries especially when boundaries are diffused. In the paper, we illustrate our ideas on two different segmentation algorithms namely, seeded region growing [1] and region competetition [10,6]. Several examples will be presented to illustrate the effectiveness of the proposed algorithm.

2 Watershed Transforms: Overview

Watershed segmentation [8,5] is a morphological gradient-based technique, which can be intuitively described as follows: View the gradient image as a height map, and gradually "immerse it in water", with water leaking through the minimum gradient points, rising uniformly and globally across the image. Place a "dam" when two distinct bodies of water (catchment basins) meet and continue the process until water has reached all the points of the image. The dams provide the final segmentation. This can be interpreted in the image domain as the growth of seeds placed on the minima of the image gradient height map at a time proportional to their height that finally converges on the crest lines of the gradient map. This is a powerful approach especially where local gradients cannot be defined, *e.g.* diffused edges. Since most structures contain several catchment basins on them, a typical watershed segmentation produces a large number of regions for even simple images, which is known as the *over-segmentation* problem. Many regions can be effectively reduced via nonlinear smoothing filtering [9,2]. The rest of them can be grouped together via region-growing techniques [8] or using marker methods [8,4].

Recently, Stoev and Strasser [7] proposed a technique for local computation of watersheds via simulating the rain-falling process. While this algorithm may work well on certain images, it cannot fill every basin correctly. Specifically, errors occur at places where basins have necks, or protrusions, *i.e.* water may not flow into the correct basin during the raindrop tracking process, Figure 1. Our implementation of this algorithm has proved the existence of this problem and the basins extracted from this algorithm appear *blocky* due to missed protrusions and necks in basins. In fact, Vincent and Soille [8] also discussed the errors that arise in tracking the path of a raindrop in digital image domain.

[1] Region and basin terms are interchangeably used throughout of this paper.

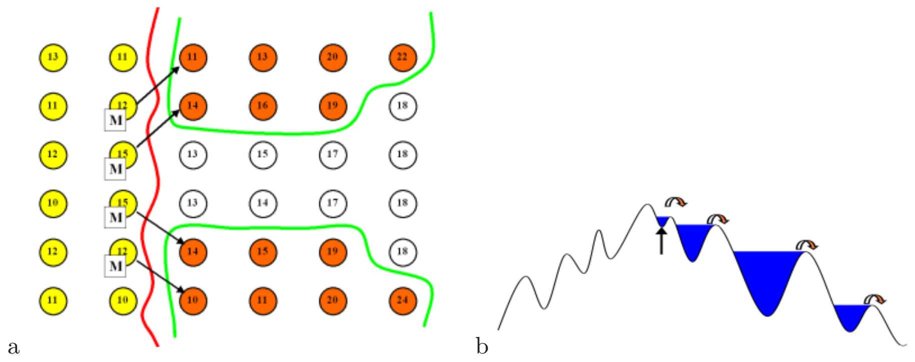

Fig. 1. (a) Breadth-First Problem: Water cannot flow through narrow regions in the breath-first type basin-filling algorithm since flooding on the dam (at the maxima) is stopped. Arrows show the pixels, which cause maxima to stop (b) Depth-First Problem: Whenever water reaches a dam, it starts filling the neighboring region from its minimum. The same process is applied to all neighboring regions iteratively, resulting in exploring too many regions if the minima of neighboring regions are in monotonocally decreasing order.

3 Local Watershed Operators

In this paper, we propose a new algorithm for computing watershed transform of an image *locally*. It is based on filling a region from a user selected point. The main goal is to fill a basin and compute its boundaries correctly and locally. This goal is satisfied via filling the main region and its immediate neighboring regions *simultaneously*. This *three-layer basin filling* algorithm, is based on the following two filling techniques.

Breadth-First Basin Filling: The first and most obvious approach of basin extraction is based on filling it with water and building dams at its ridges. First, the minimum of a user-selected region is computationally determined with gradient descent algorithm. Then, the region is filled with water starting from the minimum. When the water level reaches the ridges (watershed lines), dams (called maxima in our representation) are constructed to stop the water flowing into the neighboring regions. When a region is surrounded by dams, the filling process terminates. Second, the minimum of a neighboring region is determined from the dams (maxima points), again with a gradient descent algorithm. The same filling and construction of dams approach is recursively applied to fill the neighboring basins. Once the neighboring regions are filled, they can be checked for merging to the main region via segmentation criteria.

We have implemented the above algorithm. The filling process is implemented via region growing type operators, *i.e.* pixels visit eight-neighborhood, and bucket-based queueing for computational efficiency. Queueing is necessary for simulating the water rise. Specifically, we start from the minimum point and visit its eight neighbors and put them into buckets based on their height function. Then, the pixel with the minimum value is removed from the bucket for further growing. First, we check if any neighbor of this pixel has a lower

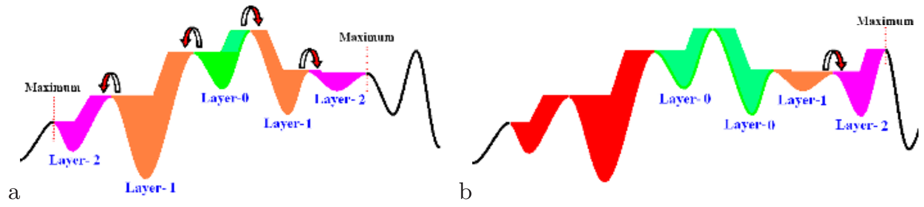

Fig. 2. The three-layer representation of basins during flooding (a) and the update of the layers after merging and continued flooding (b).

intensity value. If there is such a pixel, it means that we are in the vicinity of a watershed line and we mark that point as maximum, and its neighbors are not inserted into buckets. This growth process continues until no more pixels are left in the buckets, *i.e.* the basin is filled and surrounded by maxima points. As illustrated in Figure 1a, this filling process prevents passing through narrow regions, necks or protrusions, thus, basins cannot be correctly determined. The neck problem in this algorithm may be solved by using differential operators, such as second order derivatives, however, they are sensitive to noise and filter sizes. Our goal is to have a robust basin filling algorithm with no higher order gradient computations.

Depth-First Basin Filling: The above algorithm builds dams wherever it sees a lower height during the filling process. The next possible approach for computing local watershed transform is based on filling the neighboring regions whenever water flows into them instead of building dams. Specifically, when a pixel visits a neighboring pixel with a lower intensity, the minimum of the neighboring region is computed and water level is reduced to its minimum. The filling process continues from the new minimum until the user selected basin is filled, Figure 1b. During this process, the pixels where water from different basins are visited more than once are marked as *watershed-pixels* [2] as in [8]. The algorithm works very well if the selected basin is surrounded by regions with higher minima. However, when a region neighbors several regions with lower minima, which in turn also neighbors other regions with lower minima, the algorithm will fill considerably a large number of regions. In our experiments, we have observed that the algorithm can easily explore more than half of an image. Thus, this algorithm cannot be used where the local growth process is extremely important.

Three-Layer Basin Filling Algorithm: We now present a *three-layer basin filling* watershed algorithm, which combines the two approaches mentioned above. In this algorithm, the user selected region of interest is assigned to *layer-zero* and its immediate neighboring layers are marked as *layer-one* while the immediate neighbors of layer-one regions are marked as *layer-two*, Figure 2.

It is already shown that a basin and its watershed lines can be correctly computed if its neighboring regions are filled simultaneously, *i.e.* they reach the ridges at the same time. It is enough to start (or continue) the filling process

[2] Watershed-pixels are equidistant to two or more regions and are important in watershed implemetations. Watershed line is a more general term for describing ridges.

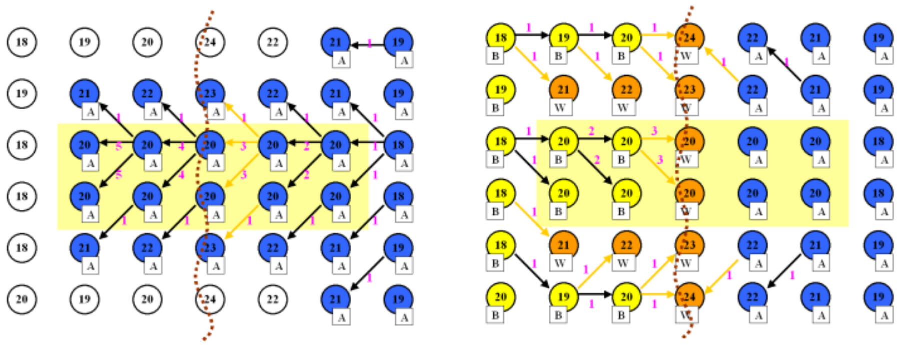

Fig. 3. The partitioning of plateau (level 20) between two basins A and B seperated by dashed line: Some pixels next to plateau are marked as watershed-pixels(W) incorrectly. They are corrected by using the source information. The propogation of distance is indicated by arrows.

right before the lowest level ridge they share. Then, the filling algorithm simulates region competition between these two regions and correctly constructs all watershed lines the regions share. This property is used in our algorithm between layer-zero vs. layer-one, and layer-one vs. layer-two regions. In other words, layer-zero and layer-one regions initialize the neighboring regions whenever they reach to the first watershed line that they share with their neighbors, thus allowing simultaneous filling and correct determination of watershed lines between these layers. On the other hand, layer-two regions build dams wherever they see new regions. Thus, it is possible that layer-two basins cannot be fully filled due to the neck problem. The three-layer filling process stops when layer-zero and all layer-one regions are completely filled, *i.e.* all watershed lines have been formed.

Watershed algorithms apply special processes when water reaches plateau, a local flat region. A plateau does not introduce any problems if it is totally inside a basin. However, when it is located between two or more basins, extra care must be taken to partition the plateau between the regions correctly. Like traditional watershed algorithms, the distance transform [3] is used in our algorithm to partition the plateaus between regions. Specifically, assume that a pixel P_i visits its neighboring pixel P_j, then the distance value at P_j is defined as

$$dist(P_j) = \begin{cases} 1 & \text{if} \quad g(P_i) < g(P_j) \\ min(dist(P_j), dist(P_i) + 1) & \text{else if} \quad g(P_i) = g(P_j) \end{cases} \tag{1}$$

where $g(P)$ is the height function at P. The above equation uses a simple distance metric. More accurate distance metrics, such as Euclidean can be used as well, however, at the expense of more computational time.

The distance transform allows the correct partitioning of plateaus between two regions. However, when a layer-zero (or layer-one) initializes a region from a plateau, a special case occurs, as in Figure 3. In the figure, region A propagates on the plateau (level 20) until it detects a new region B, *i.e.* it sees an empty lower level (level 18). During the propagation, all plateau points and the pixels next to the plateau are marked as region A. When water is initialized from B, it corrects the pixels at the plateau via distance transform. However, pixels next to the plateau are marked as watershed-pixels because collision from regions A and B are marked at these pixels. Specifically, region B visits these pixels but it cannot change their region labels because it brings a distance value of one as region A does. This error is corrected when the pixels from watershed-pixels are selected from the bucket. Specifically, if their sources are different, then they were correctly marked as watershed pixels; otherwise their correct regional labels are assigned to them and water starts rising from them.

Once the layer-zero and layer-one regions are filled correctly, *i.e.* no pixels exist in the buckets, the filling process terminates. Now, region merging can be applied between the layer-one basins and the layer-zero basin. When a layer-one region is merged to the layer-zero basin via some merging criterion, e.g. thresholding, it is also marked as layer-zero and the regional information of layer-zero is updated accordingly, Figure 2. In addition, all the neighboring layer-two regions of the merged layer-one region are updated to layer-one status. After merging, the filling process is restarted from the minimum point of the previous level-two regions' maxima list. This may cause some parts of layer-two regions to be processed again, but it is important to lower the water level to the place where the first maximum was marked. The lowering of water level to the first maximum point allows the algorithm to initialize the neighbors of the previous level-two regions(converted to layer-one after merging) as the new level-two regions, thus, avoiding the neck problem, Figure 1. The filling process continues until there are no more pixels left in the buckets, *i.e.* all new layer-one regions are also filled. The algorithm continues by merging, updating and flooding until a user-defined convergence criterion or a combination of several criteria are satisfied.

In this section, we proposed a new method for computing watershed transforms locally and correctly. The new algorithm consists of filling and merging processes. We call them watershed operators. Similarly, a deletion operator can be easily defined. We believe that these watershed operators, namely filling, merging and deleting can be used as basis for the segmentation processes, which will be described in the next section.

4 Medical Image Segmentation via Watershed Operators

In most local segmentation algorithms, the growing/deleting operations are done on pixels. However, we propose that regions can be used instead of pixels. Specifically, we illustrate the ideas on seeded region growing [1] and region competition algorithms [10,6].

Combined with morphological operations, seeded region growing can be considered as one of the most practical segmentation algorithms in medical image applications due to its simple implementation and near real-time execution. Typically, from a user selected point, pixels are added to the growing region until a threshold criterion is no longer satisfied. This method works well if the regions are rather well isolated and a good threshold value is determined. However, region growing algorithms cannot localize edges, especially when boundaries are

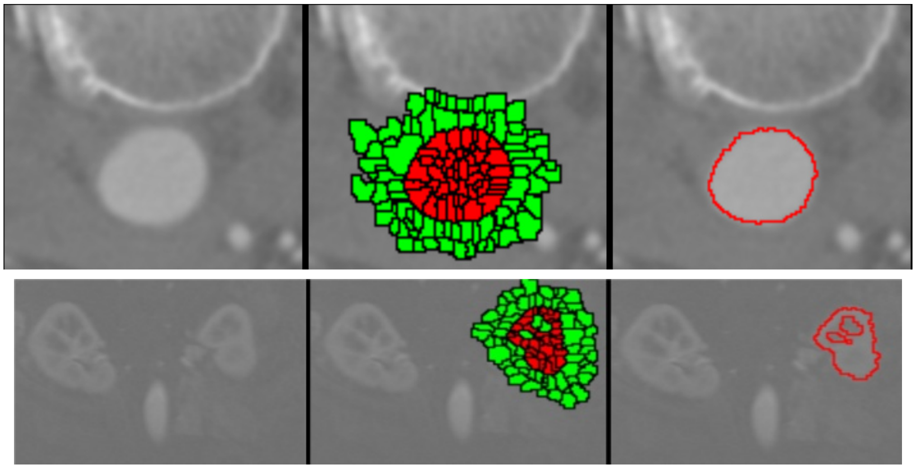

Fig. 4. Results of seeded region growing: The segmented region is red while other explored regions are green and watershed lines are black. (Top) Segmentation of aorta from CTA in orthogonal view. Seed point is given by the user (or from vessel centerline model) inside the aorta. (Bottom) Partial segmentation of a kidney from MRA. Again, user enters the seed point on the part of kidney.

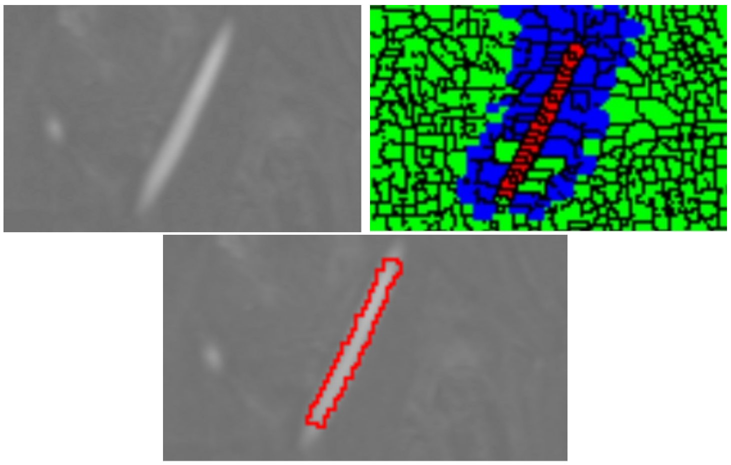

Fig. 5. Result of region competition: One seed is given in the vessel, one seed is given in the background. Red represents segmented vessel. Green represents the background regions. Blue represents the regions, which are first taken by the vessel, but then lost to background after competition.

diffused. In this paper, we propose that region growing can be implemented via watershed operators. Instead of pixels, basins are added to the growing region. We have implemented this algorithm and the threshold is determined from a bleeding criterion. Specifically, the bleeding criterion monitors the size of the growing region and the histogram of the segmented region. If a newly added basin changes the histogram and the number of pixels in the segmentation significantly, bleeding occurs and this stops the propagation. The main advantage of this algorithm is its ability to better localize edges even in the case of diffused edges. Figure 4 presents the segmentation of vessels and parts of the kidney by this algorithm.

The second example of applying watershed operators for image segmentation is the region competition [10]. In general, region competition works well, but it cannot guarantee detecting the boundaries at the edge point since it is rather difficult to incorporate edge information to this process, and it requires advanced deformable model evolution [6]. We have implemented a rather simpler version of region competition via watershed operators. Specifically, in our implementation, the growing regions compete for basins via watershed operators, namely, filling, merging and deleting, and produce well localized boundaries. The result of this region competition algorithm is also illustrated in Figure 5.

5 Conclusion

In this paper, we presented a three-layer basin filling algorithm for computing watershed transforms locally and accurately. We expect that the proposed algorithm will be important in the semi-automatic segmentation of medical structures in large data sets, *e.g.* CTA data obtained from multi-detector CT machines. In addition, we proposed that these watershed operators can be efficiently used in segmentation algorithms.

References

1. R. Adams and L. Bischof. Seeded region growing. *PAMI*, 16(6):641–647, 1994.
2. E. B. Dam and M. Nielsen. Nonlinear diffusion schemes for interactive watershed segmentation. In *MICCAI*, pages 596 – 603, 2000.
3. P. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.
4. R. J. Lapeer, A. C. Tan, and R. Aldridge. Active watersheds: Combining 3D watershed segmentation and active contours to extract abdominal organs from MR images. In *MICCAI*, pages 596 – 603, 2002.
5. L. Najman and M. Schmitt. Watersheds of a continuous function. *Signal Processing*, 38(1):99–112, 1994.
6. T. B. Sebastian, H. Tek, J. J. Crisco, S. W. Wolfe, and B. B. Kimia. Segmentation of carpal bones from 3d CT images using skeletally coupled deformable models. *Medical Image Analysis*, 7(1):21–45, 2003.
7. S. L. Stoev and W. Strasser. Extracting regions of interest applying a local watershed transformation. In *IEEE Visualization*, 2000.
8. L. Vincent and P. Souille. Watersheds in digital spaces: an efficient algoritm based on immersion simulations. *PAMI*, 13(6):583–598, 1991.
9. J. Weickert. Review of nonlinear diffusion filtering. In *First International Conference, Scale-Space*, pages 3–28, Utrecht, The Netherlands, 1997. Springer.
10. S. C. Zhu and A. L. Yuille. Region competition: Unifying Snakes, Region growing, and Bayes/MDL for multiband Image Segmentation. *PAMI*, 18(9):884–900, 1996.