Scanning Order Dependencies in Watershed Transforms

Bryan J. Mealy

UCSC-CRL-02-37
December 12, 2002

Computer Engineering Department
University of California
Santa Cruz, CA 95064

Abstract

Immersion simulation-based Watershed Transforms (WSTs) generally contain two types of implementation dependent ordering operations. In this paper, we examine the scanning order dependencies present in an immersion simulation-based WST algorithm. The WST algorithm examined generates a complete partition of the original image without pixels being designated as watershed pixels. We describe the ordering dependencies and analyze how they affect the final partition when different scan orders are applied.

The results of our analysis indicate the difference in the partitions generated from using different scanning orders is on the order of one pixel per region. Moreover, these differences comprise of a relatively small portion of the total pixel count for the test images used. The number and location of these differences are such that they are not detectable when visually comparing the region boundaries of partitions generated using different scanning orders.

Keywords: watershed transform, immersion simulation, initialization scanning order, neighborhood scanning order.

1 Introduction

A variety of methods are used to implement Watershed Transforms (WSTs). Flooding algorithms, or immersion simulations, are widely considered the fastest and most accurate WST algorithms and are used in a majority of published WST applications. In this paper, we examine the immersion simulation-based algorithm of Buecher and Meyer [1] in conjunction with the preprocessing algorithm described by Dobrin *et al.* [2].

WST algorithms generate one of two types of outputs: a complete image tessellation or an image containing regions delineated by pixel-wide watershed lines. The immersion simulation-based WST algorithms appearing the literature generally have the same structure and give similar results. They do, however, differ in the type of output they generate. Most immersion simulation WST algorithms generate watershed lines in addition to regions [2, 4]. These watershed lines only serve to delineate regions but are not members of regions. The WST algorithm of Buecher and Meyer [1] does not generated watershed lines; region boundaries are delineated by the edge between two pixels of different regions.

Immersion simulation-based WST contain two types of scanning order dependencies: neighborhood scanning orders and initialization scanning orders. Neighborhood scanning order refers to the sequence in which pixels are examined during WST implementation. Initialization scanning order refers to the method in which catchment basins are first formed and listed by the WST algorithm. Scanning order dependencies cause pixels in the generated partition to be placed in different regions when different scan orders are applied.

2 Immersion Simulation WST Algorithm Analysis

Fig. 1 shows the four steps required for an application of an immersion simulation-based WST. This diagram shows three steps of preprocessing followed by the actual WST. The first step generates a gradient image based on the original grayscale image. The next step designates certain portions of the image as regional minima which become the *prick-points* for the ensuing immersion simulation. The initial pixel queuing step adds pixels to the *ordered queue* to create a starting state for the WST. The final step is the actual WST.

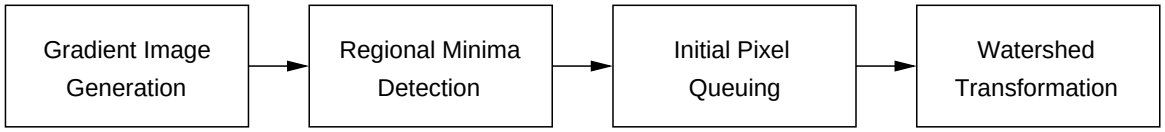

Figure 1: Process flow of the immersion simulation-based WST algorithm.

In this section, we examine the uniqueness of the partition generated by the WST. Moreover, we describe the effects that the blocks of Fig. 1 have on the generated partition. We analyze each step in order to determine the dependencies the steps have on the uniqueness of the generated partition. Implementation details are provided where appropriate but we assume the reader

is familiar with the concepts of immersion simulation-based WSTs. Reference [4] provides an introduction to immersion simulation-based WSTs.

2.1 Gradient Image Generation

The first step in WST processing generates a gradient image based on the original grayscale image. Each pixel in the grayscale image, I, is mapped to a value representing the discrete gradient magnitude at its location. This mapping forms the gradient image X, *i.e.*, $\nabla f : I \rightarrow X$. Each pixel p in I is mapped to X by the function ∇f which approximates the gradient magnitude at p.

Published WST implementations generally agree that partitions generated by WSTs are not particularly sensitive to the method used to approximate gradient values. Generating the gradient image essentially provides preprocessing to the regional minima detection step. If the method used to approximate gradient values is applied equally to every pixel in the image, no scanning order dependencies are generated. We therefore shift our focus to the second block of Fig. 1.

2.2 Regional Minima Detection

The application of the WST requires locating and labeling regional minima in the gradient image. Regional minima represent the so-called prick-points used in immersion simulation-based WSTs. The partition produced by the WST is dependent upon the method used to locate and list the prick-points. The prick-points serve as seeding points for the region growing, or flooding process. Each prick-point spawns the growth of a catchment basin, and each catchment basin contains only one prick-point. In this paper, we limit our discussion to automatic[1] regional minima detection techniques.

2.2.1 Dobrin's Regional Minima Detection Algorithm

The first work by Dobrin *et al.* [2] presented several preprocessing steps prior to the WST application. This preprocessing included an algorithm for regional minima detection. Regional minima are not explicitly assigned by Dobrin's regional minima detection algorithm. In the Dobrin *et al.* approach, the algorithm instead locates and marks pixels that do not fit the definition of local minima. These pixels are labeled as *not a regional minimum*, or *NARM*s. The gradient image, X, is transformed into an image where each pixel is either a NARM or the initial value assigned by the algorithm. Pixels containing initial values are then considered regional minima. This procedure creates a "sea" of NARMs interspersed with "islands" of regional minima. This definition of regional minima renders the analogy of "prick-points" from immersion simulation models misleading since the minima are not constrained to being single pixels, or "points". Instead, a single regional minima can be as small as a single pixel or as large

[1] In this paper, the term *automatic* refers to techniques that do not require intervention from an outside user.

as a major portion of the image. WST algorithms generally use the term *markers*[2] synonymously with regional minima.

2.2.2 Uniqueness of the Marker Image

In this section, we examine the marker detection portion of the Dobrin *et al.* preprocessing algorithm. We show the marker selection by this algorithm produces a set of markers that are unique and consistent relative to a given gradient image. Moreover, we show this algorithm contains no scanning order dependencies that could affect the WST-generated partition.

To facilitate description, images are represented as numerical functions. Let D_X represent the domain of a 2-dimensional image X where $D_X \subset \mathcal{Z}^2$ and $\mathcal{Z}$ is an integer. Pixel values form the range of X which comprise the set $R = \{0, 1, \ldots, N-1\}$, where N is a positive integer. Accordingly, $X : D_X \rightarrow R$ with the value of X at any pixel $p \in D_X$ denoted by $X(p)$. The underlying local digital grid is denoted by G with $N_G(p) = \{p' \mid p' \in G\}$ defining the local pixel neighbors[3] of any pixel p. Any pixel p' is considered to be adjacent to, or a neighbor of, pixel p if $p' \in N_G(p)$.

> **Definition 1**: A regional minimum M of X at a height h is a connected set of pixels P with $|P| \geq 1$, from which it is impossible to reach a pixel p' with $X(p') = h'$ and $h' < h$, without encountering pixel p'' with $X(p'') = h''$ where $h < h''$.

In general, a marker selection algorithm examines every pixel in the gradient image to determine if the criteria stated in Definition 1 is met. A straight-forward approach to marker selection would be to compare each pixel value to the value of its neighbors using raster scan order. The algorithm of Dobrin *et al.*, however, applies Definition 1 to each pixel in the image recursively. Moreover, the algorithm recurses through connected NARMs; the recursion is halted and transferred to the next NARM in the image once all the pixels in the current NARM are processed. There is no time advantage to the approach of Dobrin *et al.* and actually has increased space requirements[4] compared to a non-recursive approach.

Marker detection depends solely upon the value of a pixel relative to the value of adjacent pixels. The relationship between pixels and neighboring pixels is independent of the order in which pixels are considered. This independence allows Dobrin's marker detection algorithm to generate the same marker set regardless of the processing order applied. Therefore, the region minima detection algorithm of Dobrin *et al.* contains no scanning order dependencies. This result can be shown experimentally by applying the regional minima detection algorithm to an image and versions of the same image rotated by 90, 180, and 270 degrees. The rotated images provided a simple method to effectively apply different scanning orders. Results of marker detection are compared after rotating the images back to their original orientations. As

[2]The term *markers* is sometimes used to signify areas that were interactively selected by an outside user.

[3]In regards to WSTs, we consider only an 8-connected pixel model.

[4]The increase in space requirements is dependent upon image characteristics. Image areas containing higher spatial frequencies are generally associated with high counts of contiguous NARMS which deepens the recursion.

expected, the image are identical. Example images from these experimental results are trivial and are not presented here.

2.2.3 Neighborhood Scanning Order Dependency

Processing in immersion simulations proceeds by examining and queuing pixels. When a pixel is processed by the algorithm, the neighbors of that pixel are examined. The order in which these local pixels are examined is referred to as the neighborhood scanning order. This raises a question of whether the order in which neighboring pixels are processed have any effect on the final regions generated by the WST?

This question was considered by Dobrin *et al.* in [3]. In this work, Lemma 4 states: *The neighborhood scanning order does not affect the result of Meyer1 and Meyer2 algorithm.* A complete proof and further description of this lemma is found in [3]. Therefore, no neighborhood dependencies are present and the second block of Fig. 1 does not affect the uniqueness of the final partition.

2.3 Initial Pixel Queuing

WST processing up to this point has not generated scanning order dependencies. We now examine how the *initialization scanning order* affects the final WST partition. Initialization scanning order refers to the order with which the marker image is scanned when searching for unlabeled markers[5]. Pixel queuing is a process inherent to every immersion simulation-based WST. Initial pixel queuing occurs once a marker is encountered and labeled. The first pixels queued are the pixels surrounding the markers. The order in which pixels are queued is therefore dependent upon the order in which markers are encountered.

In the Dobrin *et al.* algorithm, processing of the ordered queue does not occur until each marker is encountered and the pixels surrounding the marker are queued. This approach depends upon some pre-defined order in which to process the pixels surrounding regional minima. Since the regional minima are explicitly detected, a scan of the marker image is required to initiate the queuing of pixels surrounding the markers. Dobrin's preprocessing algorithm does not state a search order for marker discovery and is therefore dependent upon the initial pixel queuing order.

Initial pixel queuing occurs after regional minima detection and is embedded in the marker labeling portion of the algorithm. The first pixels to be placed on the ordered queue are the NARM pixels neighboring the markers in I_M. These pixels are denoted by $\beta(M_i)$ with $\beta(M_i) = \{p \,|\, I_M(p) \neq m_i \text{ and } \exists\, p' \in N_G(p) \text{ with } I_M(p') = m_i\}$ where $i = \{1, 2, \ldots, k\}$ and k is the number of markers in I_M. It is obvious from this definition that the order in which pixels appear on the ordered queue are dependent upon the scanning order used to locate unprocessed markers.

The initialization scanning order dependencies inherent in all WSTs are attributed to the analog to digital conversion of images. The fact that these dependencies consequently cause differences to appear in the regions of the final partition was noted and described in an analysis

[5]Rasterscan order is considered a initialization scanning order.

of watershed algorithms by Dobrin *et al.* [3]. Dobrin's work traces the dependency of the final watershed partition to cases where flat plateaus of pixels are found between catchment basins. As the flooding progresses, the rising waters become biased toward the basin that is processed first. Basin processing order is determined by the initialization scanning order. This biasing effect only occurs when an odd number of pixels separate the two catchment basins in question. The same effect exists with WST algorithms generating pixel-wide boundaries but occurs in cases where an even number of pixels separate catchment basins.

The biasing effect is best described in the context of initial pixel queuing since this is the first time it can occur. The Dobrin work centered the discussion on the general case described in the previous paragraph where an odd number of pixels with symmetric values separate two catchment basins. In the general case, the bias toward one basin does not emerge until pixels adjacent to two regions are assigned to a catchment basin. Fig. 2 demonstrates the initialization scanning order dependency in the context of initial pixel queuing. Fig. 2(a) shows the original image with numbers representing the grayscale values. Fig. 2(b) is the associated marker image with two markers, M_a and M_b, shown with shaded pixels and labels **a** and **b**, respectively. Fig. 2(c) and Fig. 2(d) are the partitions generated with a left-to-right and a right-to-left initialization scanning order, respectively. Pixels in the different catchment basins are indicated with labels **a** and **b** in Fig. 2(c) and Fig. 2(d). The hatched lines highlight the differences between the resulting catchment basins. In this image, pixels not in M_a or M_b are contained in $\beta(M_a)$ or $\beta(M_b)$. This simple example demonstrates the dependency the WST generated partition has upon initial pixel queuing. For this example, the differences in the final partition are the line of pixels separating the two regions, *i.e.*, $\beta(M_a) \cap \beta(M_b)$.

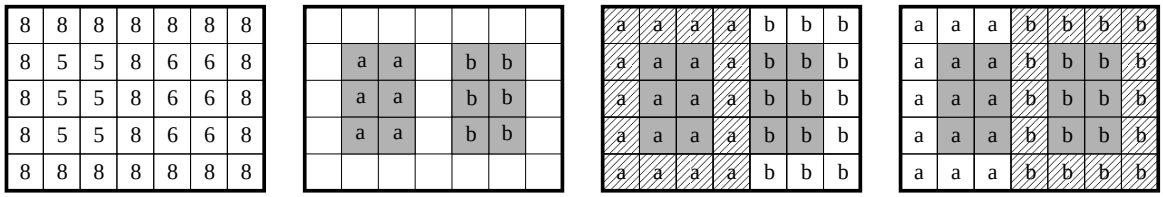

Figure 2: An example demonstrating the WST generated partition's dependency upon initialization scanning order. (a) is a grayscale image and (b) is the associated marker image with two markers labeled **a** and **b**. (c) and (d) show the final partitions generated using two different initialization scanning orders.

2.4 The Watershed Transform

The final block of Fig. 1 is the actual application of the Watershed Transform. Previous analysis showed that scanning order dependencies exist prior to the application of the WST. More specifically, at this point in WST preprocessing, marker pixels and the surrounding pixels are assigned to regions. Changing the scanning order would possibly place these pixels in

lena (256x256) tripod (256x256) peppers (256x256)

Figure 3: The test images.

other regions. The WST application further promotes the differences appearing in the partially segmented image as water levels rise and regions are grown. Moreover, the WST essentially acts upon the pixel differences already present in the image. The further increase in pixel differences that occur in the WST step are are due to the mis-classifications present prior to the WST and not the WST itself. This result is quantified in the next section.

3 Results

As described in the previous section, the exact form of the partition generated by a WST is dependent on the initialization scanning order. Fig. 3 show the images used for testing. Fig. 4 shows the differences when four different initialization scanning orders are used. The four initialization scanning orders are generated by applying WST processing to an image and to three rotated versions of the same image. The non-rotated version is considered to be the baseline case. The other three scanning orders are effectively generated by applying WST processing to 90, 180, and 270 degree rotations of the original image. The 90, 180, and 270 degree rotations are referred to as UR-LR, LR-LL, and LL-UL, respectively.

Fig. 4 demonstrates the effects of initialization scanning order. Fig. 4(a)-(c) show the region boundaries after the WST is applied to the test images. The images in Fig. 4(d)-(i) are based on the worst-case scanning order (LR-LL) of Table 1. Fig. 4(d)-(f) show the pixels enumerated by the fourth column and LR-LL row of Table 1. Fig. 4(g)-(i) show the pixels enumerated by the sixth column and LR-LL row of Table 1. These examples show that although there are differences, they are seemingly negligible. Running this experiment on other test images generated similar results but are not shown here.

Table 1 lists results from applying the different initialization scanning orders. The second column lists the number of regions in the partition. Column four lists the number of pixels that are in different regions than the baseline case after initial pixel queuing. Column five shows the percent differences of column four based on the total number of pixels in the image. The sixth column lists the number of pixels in different regions after the WST is complete. Columns seven shows the percent difference of column six based on the total number of image pixels. The final

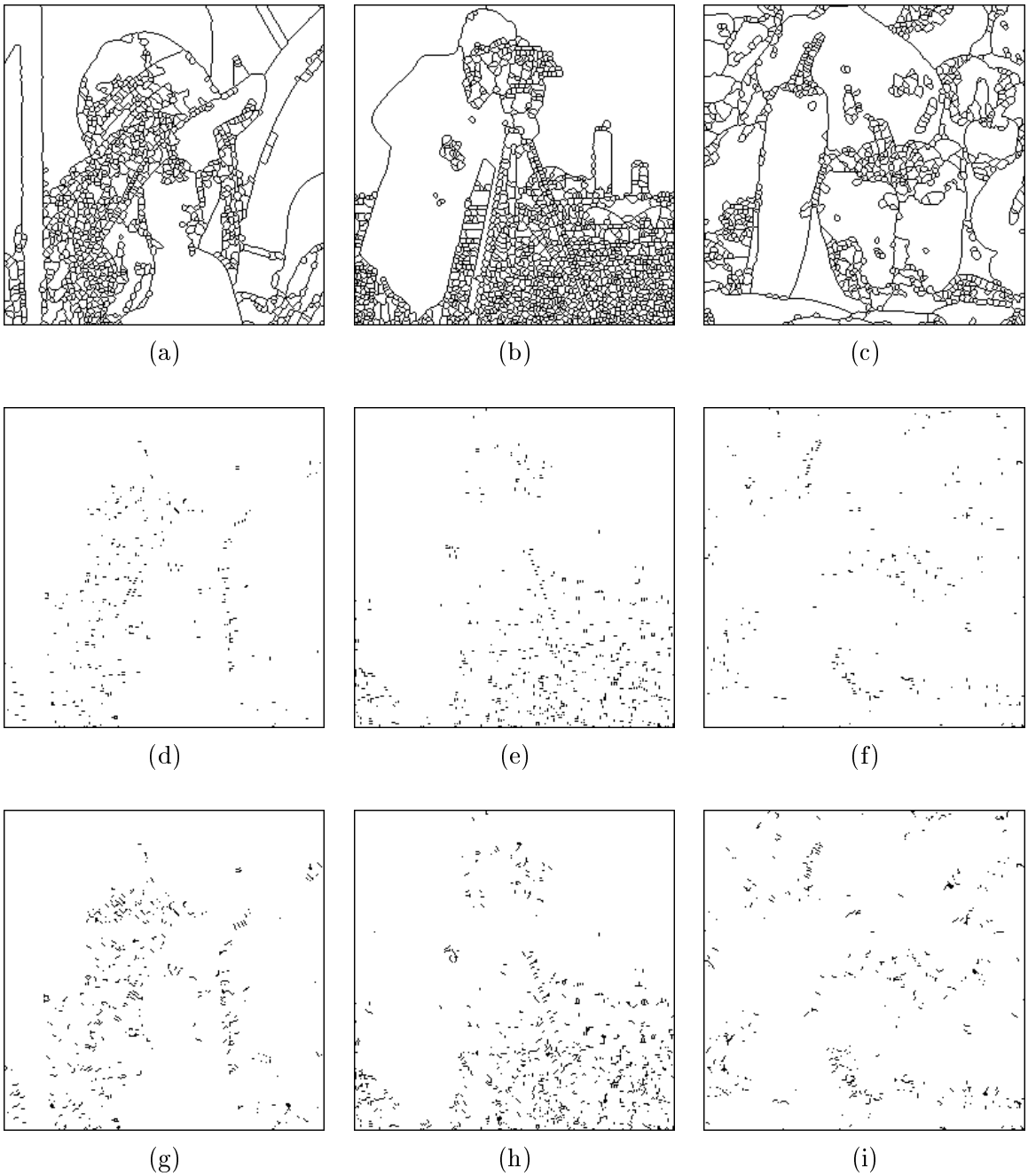

Figure 4: An example showing the effects of initialization scanning order on the WST partition. Results are listed for the LR-LL scan order. (a)-(c) show the image boundaries for the baseline scan order. (d)-(f) show the pixel differences compared to the baseline scan order after the initial pixel queuing. (g)-(i) show the pixel differences in the final watershed partition.

			Initial Pixel Queuing		Watershed Partition		
image	# regs	scan order	# diffs (pix)	% total of image	# diffs (pix)	% total of image	# diffs per region
lena	859	UR-LR	416	0.64	796	1.22	0.93
		LR-LL	547	0.84	1008	1.54	1.17
		LL-UL	242	0.37	438	0.67	0.51
tripod	1343	UR-LR	538	0.82	894	1.36	0.67
		LR-LL	1023	1.56	1628	2.48	1.21
		LL-UL	614	0.94	987	1.51	0.73
peppers	751	UR-LR	329	0.50	655	1.00	0.87
		LR-LL	460	0.70	890	1.39	1.19
		LL-UL	240	0.37	489	0.75	0.65
AVGS	984		490	0.75	865	1.32	0.88

Table 1: Results of applying different initialization scanning orders to the test images.

columns lists the number of pixel differences in the final partition per number of regions in the image.

4 Conclusion

Our results demonstrate that the WST's dependency on scanning order is relatively small. The differences in the partitions generated from using different scanning orders is less than one pixel per region (0.88) when all image regions are considered. These differences are found in the higher frequency areas of the image and exhibit qualities of randomness. Moreover, initialization scanning order differences after the initial pixel queuing step average less than one percent (0.75%) of the total number of image pixels. This initial number of pixels placed in different regions approximately doubles (1.32%) in the final partition after the WST is applied. Despite their number and location, differences in regions are not visually detectable when the region boundaries of partitions generated using different scanning orders are visually inspected.

References

[1] S. Beucher and F. Meyer. The Morphological Approach to Segmentation: the Watershed Transformation. In *Mathematical Morphology in Image Processing*, pages 443–481. Marcel Dekker, 1993.

[2] B. Dobrin, T. Viero, and M. Gabbouj. Fast Watershed Algorithms: Analysis and Extensions. In *Proc. Nonlinear Image Processing V*, volume 1769, pages 209–220. SPIE, February 1994.

[3] T. Viero, B. Dobrin, J. Astola, and M. Gabbouj. Further Analysis of Watershed Algorithms. volume 2300, pages 331–342. SPIE, 1994.

[4] L. Vincent and P. Soille. Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 13(6):583–598, June 1991.