Block-Based Predictive Watershed Transform for Parallel Video Segmentation

Jung-Whan Jang and Hyuk-Jae Lee

Abstract—**Predictive watershed transform is a popular object segmentation algorithm which achieves a speed-up by identifying image regions that are different from the previous frame and performing object segmentation only for those regions. However, incorrect segmentation is often generated by the predictive watershed transform which uses only local information in merge-split decision on boundary regions. This paper improves the predictive watershed transform to increase the accuracy of segmentation results by using the additional information about the root of boundary regions. Furthermore, the proposed algorithm is processed in a block-based manner such that an image frame is decomposed into blocks and each block is processed independently of the other blocks. The block-based approach makes it easy to implement the algorithm in hardware and also permits an extension for parallel execution. Experimental results show that the proposed watershed transform produces more accurate segmentation results than the predictive watershed transform.**

Index Terms—**Object segmentation, watershed transform, vincent-soille algorithm, predictive watershed transform, hardware implementation**

Manuscript received Aug. 31, 2011; revised Dec. 12, 2011.
Inter-university Semiconductor Research Center, Department of Electrical Engineering, Seoul National University, Seoul, 151-742, Korea.
E-mail : hyuk_jae_lee@capp.snu.ac.kr

I. INTRODUCTION

Object segmentation, which partitions an image into multiple sets of non-overlapping closed regions, has been widely used in many applications such as video surveillance, medical diagnosis, satellite imaging and human computer interface (HCI). Among many segmentation algorithms, watershed transform is one of the effective algorithms to segment objects with unclear boundaries [1-2]. A large amount of computation required by watershed transform makes it difficult to segment objects in real-time. As the demand for real-time object segmentation in an increasing size image grows, a number of research efforts have been made to achieve real-time watershed transform.

For object segmentation in video sequences, temporal information in addition to spatial information can be effectively used for efficient extraction of objects [3-4, 11-12]. Spatial information can be used to obtain boundaries of objects while temporal information is used to track the motion of the objects. Another speed-up approach is to divide an image into sub-blocks and each block is processed by different processors. Thus, a speed-up is achieved when multiple processor units are available. In [6], parallelism is somewhat limited because non-minima plateaus shared by several blocks need to be processed sequentially. In [7], sequential scanning based watershed algorithm consists of repeated raster and anti-raster scans within the subdomains and message passing among processors.

In [8], an efficient watershed transform for video sequences, called *predictive watershed transform*, is proposed to speed up object segmentation by identifying image regions that are different from the previous frame

and performing object segmentation only for those regions. A speed-up is achieved because the computation for watershed transform is avoided for the regions that are same as the previous frame. A challenge in this predictive watershed transform is how to handle the boundary between updating regions and unchanged regions. To reduce the computational complexity, predictive watershed transform sacrifices the accuracy of object segmentation results, and consequently, often generates incorrect segmentation results. As a result, the segments generated by predictive watershed transform are often different from those produced by the original Vincent-Soille algorithm that performs segmentation for entire frames.

This paper improves the predictive watershed transform to avoid the inaccuracy of segmentation results. For each segmented region, the information about the root of the region is stored and then used to make a decision between merge and split of the two regions on the boundary of updating and unchanged regions. For effective hardware implementation, the updating region is decomposed into blocks of the same size which can be processed in parallel. The block-based watershed transform can further reduce the computation by examining the necessity of flooding operations for each block and avoiding them when they are not necessary. As a result, the proposed watershed transform produces more accurate segmentation results than predictive watershed transform, which is observed by experimental results. Furthermore, the block-based approach makes it easy to implement the algorithm in hardware.

This paper is organized as follows. The predictive watershed transform is introduced in Section II and a block-based predictive watershed algorithm is proposed in Section III. Section IV presents experimental results and Section V concludes this paper.

II. PREDICTIVE WATERSHED TRANSFORM

This section briefly introduces a watershed transform which is a popular algorithm for object segmentation. The first step of the watershed transform is to derive morphological gradients of pixels in an input image. The level of the gradient is considered as the altitude of a topological surface. Fig. 1 shows an example of the topological surface made of gradients in an image. A root

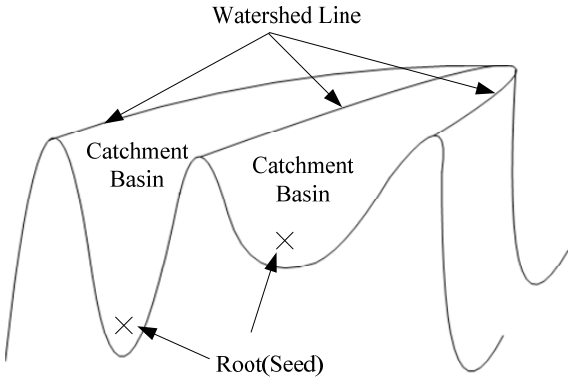


Fig. 1. Topographical surface of gradients.

is a pixel with the lowest gradient level among its neighboring pixels. Assuming that water is dropped on a pixel, gravity makes the water flows down to the lower part of the surface to eventually reach a local minimum pixel, which is the root of this surface. A catchment basin (CB) represents a set of pixels with the same root. The boundaries of catchment basins are called watershed line.

Fig. 1 shows the relationship between the root, the catchment basin and the watershed line. There may exist pixels with the same gradient levels. A plateau represents a set of pixels with the identical gradient levels [9].

Vincent-Soille watershed transform is one of the fast and effective object segmentation algorithms [10] and it consists of two major stages: sorting and flooding stages. In the sorting stage, all pixels are sorted according to their gradient levels. In the flooding stage, these pixels are visited according to the sorted level in the first stage, and then their roots are found. As a result of the flooding stage, catchment basins are formed and objects are segmented along the watershed lines.

Predictive watershed transform [3] is developed from the Vincent-Soille algorithm to speed up object segmentation for a video sequence. The main aim of the predictive watershed transform is to reduce the computational complexity by taking advantage of the temporal correlation of consecutive video frames. The four main stages of predictive watershed transform are as follows:

1. This algorithm determines the changing regions between the previous and current frames. The watershed transform is to be performed in only these changing regions.
2. Region labels and gradient values in updating area

are removed from the current frame and then morphological gradient and sorting operations are performed only for the pixels in the updating area.

3. The pixels in updating areas are then processed from the lower to the higher gradient levels, same as the original Vincent-Soille algorithm as shown Fig 2 (a). The twilled ball in the figure has Region 1 neighbor outside the updating area; therefore it is to be labeled as Region 1. Similarly, the black ball is to be labeled as Region 2. After all pixels with the labeled neighbors are processed in each level, pixels without labeled neighbors are visited and given a new label. In Fig. 2(a), the gray ball represents such a pixel and a new label (Region 3) in this example is given.
4. Finally, all pixels are labeled, and the boundaries between regions with different labels are the watershed lines of this image. Fig. 2(b) shows the watershed lines with three labeled regions segmented by the watershed lines.

Predictive watershed transform achieves a speed-up by sacrificing the accuracy of object segmentation results. As a result, the segments generated by predictive watershed transform are often different from those produced by watershed transform that performs segmentation for entire frames. Consider the case that is shown in Fig. 3. Fig. 3(a) shows a gradient surface which consists of two catchment basins, labeled 1 and 5. This surface is partitioned into three blocks, Block I, II, and III. In the next frame, the surface is changed as shown in Fig. 3(b). Thus, Block II is the region for the watershed transform to be updated. Fig. 3(c) shows the segmentation results generated by predictive watershed transform. As shown in the figure, the catchment basin labeled 5 in Fig. 3(a) is segmented into two regions, labeled 11 and 5, respectively. Note that this is an over-segmented result because the two regions labeled 5 and 11 should be merged together. This merge is not possible in predictive watershed transform because the algorithm simply compares the gradient values on the block boundary and

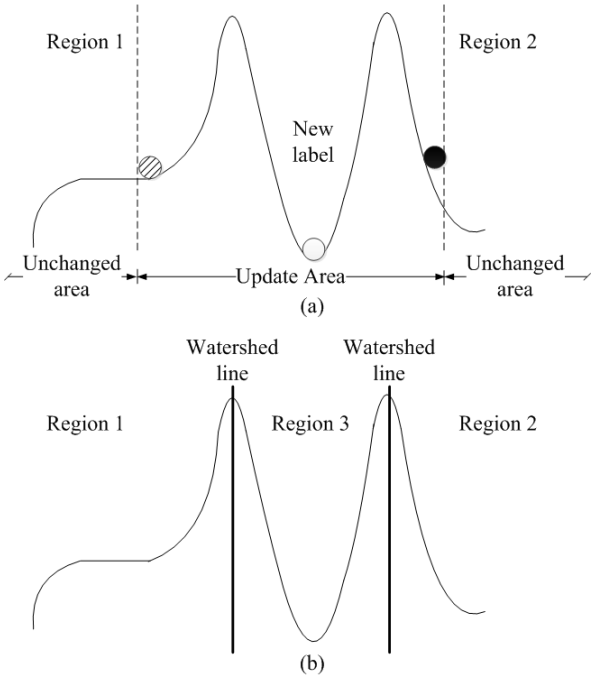


Fig. 2. Predictive watershed algorithm (a) Operation of the algorithm, (b) Segmentation results.

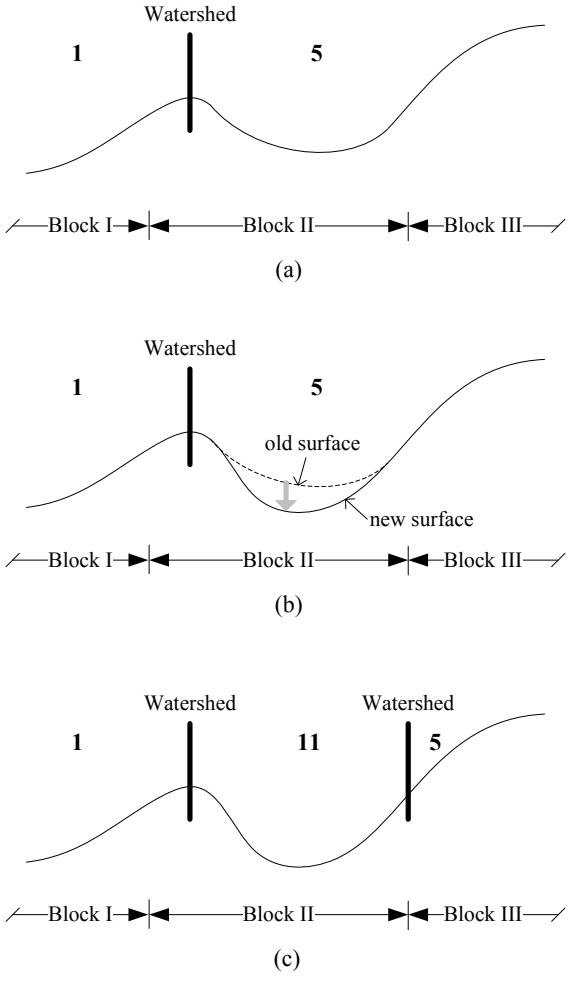


Fig. 3. Over-segmentation by predictive watershed algorithm (a) The gradient surface in the previous frame, (b) The gradient surface in the current frame, (c) Segmentation result.

always gives a new label to the root of a catchment basin inside an updated area.

III. BLOCK-BASED PREDICTIVE WATERSHED TRANSFORM FOR HARDWARE IMPLEMENTATION

1. Block-based watershed transform

For predictive watershed transform, it is necessary to find regions which include pixels that are different from those in the previous frame. In order to reduce the computational complexity in the detection of changed regions, the algorithm attempts a block-by-block examination to detect whether each block is different from the corresponding block in the previous frame. To this end, the value of a pixel is subtracted from that of the corresponding pixel in the previous frame and then the absolute values of the differences are added for all the pixels in the block. This value is called the *SAD (Sum of Absolute Difference)* of the block [13]:

$$\mathrm{SAD} = \sum_{(x,y)\in \mathrm{block}(i,j)} |\, current - reference \,| \qquad (1).$$

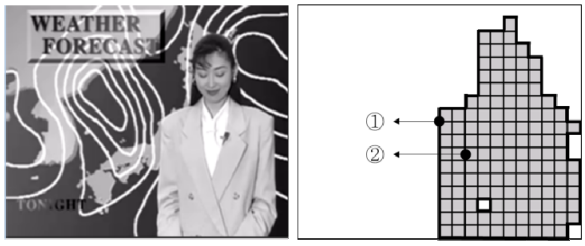
When the *SAD* of a block is greater than a predefined threshold, the block is considered as a changed block from the previous frame so that watershed segments obtained from the previous frame cannot be reused for the current frame. In this paper, the predefined threshold is chosen as 0, which implies that any slight change of pixels in the block causes watershed transform to be performed again. Let U_{flag} denote a flag that indicates whether a block is changed or not (i.e., needs to be updated or not). Then, U_{flag} is given as follows:

$$U_{flag} = \begin{cases} 1, & SAD > 0 \\ 0, & else \end{cases} \qquad (2).$$

In predictive watershed transform, all the changed regions are processed together as the original Vincent-Soille algorithm [10]. Therefore, the only modification from the Vincent-Soille algorithm is the addition of the scheme to handle the boundary between updating regions and unchanged regions. On the contrary, the watershed transform proposed in this paper partitions the changed region into small blocks and processes the watershed transform for each block independent of the other blocks. Fig. 4(a) shows an example frame of Weather sequence and Fig. 4(b) shows the regions that are changed from the previous frame. The shaded region indicates the changed region and the remaining area represents the unchanged region. As shown in Fig. 4(b), the changed region is partitioned into small blocks indicated by small squares and each block is processed independently.

The main obstacle of this block-by-block segmentation comes from the dependence among computations in different blocks. If the computation in one block depends on that in the other block, then the two blocks cannot be processed in parallel. Therefore, the block-based watershed transform proposed in this paper must guarantee independence in computations between different blocks. Another challenge in the block-based algorithm is that it needs to handle the boundary between blocks inside the updated region. This boundary is indicated by ② in Fig. 4(b). The technique is slightly different from that for the boundary between updating region and unchanged regions (indicated by ① in Fig. 4 (b)). In summary, the proposed algorithm makes two decisions for merge/split of regions across boundaries: the boundaries indicated by ① and ②. Subsection III.B presents the decision for boundary ② whereas Subsection III.C explains the decision for boundary ①. The decision for boundary ① has already been studied by predictive watershed transform and its accuracy is improved by the proposed algorithm in this paper. The decision for boundary ② is not studied by predictive watershed transform so that this paper proposes a new approach for this decision.

One of the main advantages of the block-based



(a) (b)

Fig. 4. Block-based predictive watershed transform (a) An example frame in Weather sequence, (b) Updating area partitioned into blocks.

approach is to make it easy to implement the algorithm in hardware (i.e., VLSI circuits). The Vincent-Soille algorithm includes too many global operations to be implemented in hardware. In the proposed design, each block is processed independently so that the hardware design complexity is significantly reduced. As a result, the proposed design enables hardware implementation of watershed transform. Another advantage of the block-based approach is the reduction of computational complexity by avoiding unnecessary computations for certain blocks. This second advantage is to be discussed further in Section III.4.

2. Merge/split on block boundaries

This subsection proposes an algorithm to handle the boundary between blocks inside an updating region. Block-based watershed transform often over-segments an image such that a block is transformed into different segments on the boundary of blocks even though the two blocks belong to the same object. To avoid over-segmentation, it is necessary to examine whether a segmentation boundary between blocks is a true object boundary or not. Fig. 5 shows an example with a single object placed on the boundary of two blocks: Block I and Block II. Note that the two blocks are overlapped by one column of pixels, which is used to determine whether two regions in different blocks to be merged or not. As a result of the block-based watershed transform, Region A is segmented in Block I and Region C is generated in Block II. In the previous algorithm (predictive watershed transform), only the gradient values of pixels on the boundary of two blocks are used to decide merge/split of regions across the block boundary. However, these pixel values include only local information and may lead to incorrect decision. Therefore, in this paper, the information about the root of a region is also used to make the merge/split decision. The use of the root information increases the computational complexity as well as an additional memory space. However, the information about the root is a global information which can be effectively used to improve the accuacy of merge/split decision.

Fig. 5 illustrates how to effectively use the root information for effective merge/split decision. In this figure, Root_A and Root_C represent the gradient values

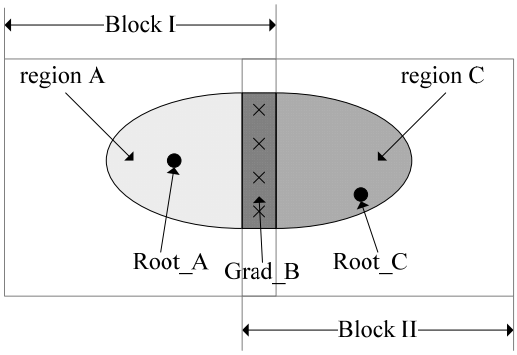


Fig. 5. Region across the block boundary.

of the roots of Region A and Region C, respectively. The conventional watershed transform generates Region A from Root_A in Block I and Region C from Root_C in Block II. Recall that the boundary between Blocks I and II is overlapped by one pixel column and the watershed results obtained from Region A and C are available on the boundary. The pixels on the overlapped column are denoted by symbol X in Fig. 5. Let Grad_B denote the gradient value of a pixel in the overlapped column. Then, Fig. 6 shows the four possible relationships among Root_A, Grad_B, and Root_C. Case I represents the case when all three values of Root_A, Grad_B, and Root_C are same whereas Case II stands for the case when the value of Root_A is smaller than those of Grad_B and Root_C which are identical. In Case III, the values of Root_A and Grad_B are the same and these values are greater than Root_C. In Case IV, the value of Grad_B is

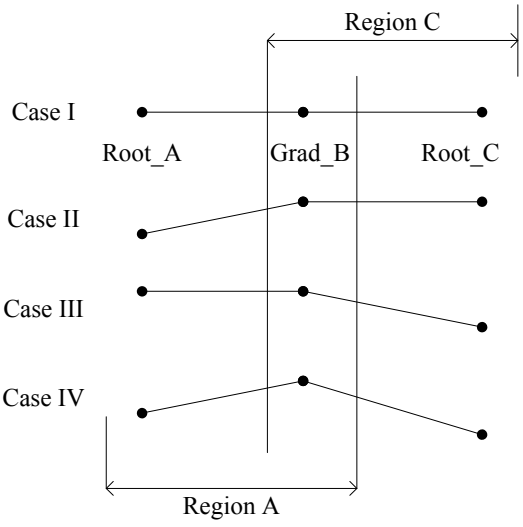


Fig. 6. Four possible relationships among Root_A, Grad_B, and Root_C.

Table 1. Decision for region merge/split

Case	*Condition*	*Decision*	*Updated root*
I	Grad_B = Root_C Grad_B = Root_A	Merge	Root_A or Root_C
II	Grad_B = Root_C Grad_B > Root_A	Merge	Root_A
III	Grad_B > Root_C Grad_B = Root_A	Merge	Root_C
IV	Grad_B > Root_C Grad_B > Root_A	Split	-

greater than Root_A and Root_C. The relationship between Root_A and Root_C does not matter in this case. Note that these are the only possible relationships among Root_A, Grad_B and Root_C because Root_A and Root_B must be always less than or equal to Grad B.

$$Root\_A \leq Grad\ B \tag{3}$$

$$Root\_C \leq Grad\ B \tag{4}$$

The classification of the four possibilities as shown in Fig. 6 allows a correct decision for the merge/split of the two regions in different blocks. Table 1 summarizes such decision. For Case I, II, and III, the two regions in different blocks must be merged whereas the two regions must be split in Case IV. If two regions are merged, the root of the merged region needs to be updated. For Case II, the new root becomes Root_A which is the smallest gradient value among Root_A, Grad_B and Root_C. On the other hand for Case III, the new root becomes Root_C. For Case I, either Root_A or Root_C can be used as the root of the merged region. For Case IV, the root update is not necessary because the two regions are not merged anyway.

Among adjacent pixels on boundary, one pixel may meet Condition I, II, or III (i.e., Merge condition) while the other pixel may meet Condition IV (i.e., Split condition). If there adjacent pixels belong to one regions in each side of the boundary, then the two regions in the both sides of the boundary are merged.

3. Merge/split decision on the boundary between the updating and unchanged regions

This subsection proposes an algorithm to handle the boundary between updating blocks and unchanged blocks. The algorithm differentiates two cases depending on whether a segment on the boundary of the updating block includes the pixel with the minimum gradient or not. When the minimum gradient is not included in the updating block, then the boundary update is exactly the same as that proposed in Section III.B. In the other case, additional consideration is necessary for correct segmentation. This case is illustrated with Fig. 7. Fig. 7 (a) and (b) are the same as Fig. 3(a) and (b), respectively.

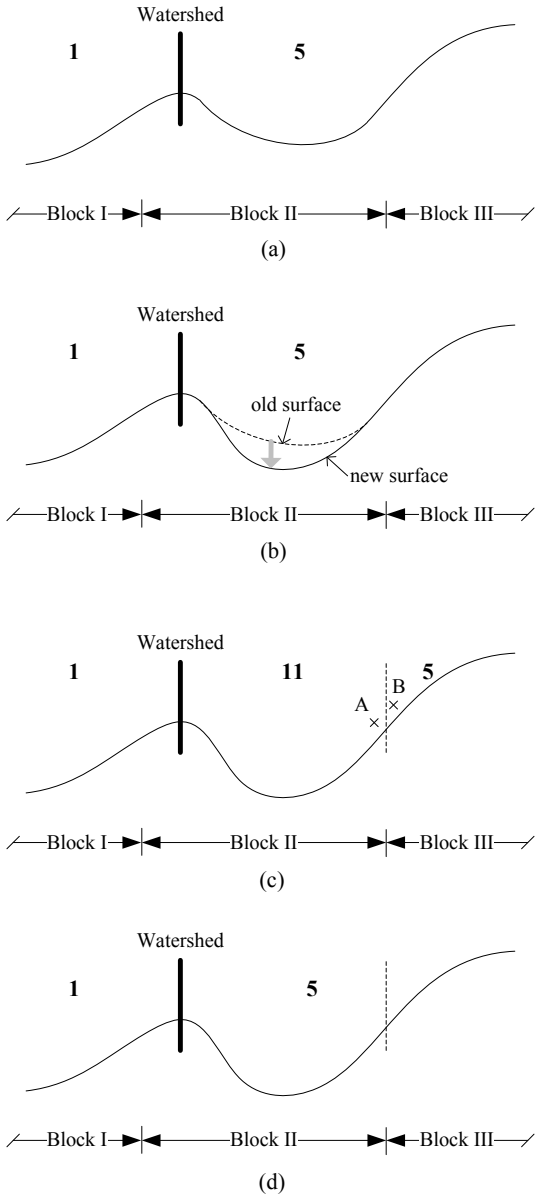


Fig. 7. Operations of the proposed watershed transform (a) The gradient surface in the previous frame, (b) The gradient surface in the current frame, (c) Merge/split decision on the boundary between Blocks II and III, (d) Segmentation result.

Recall that predictive watershed transform results in over-segmentation as shown in Fig. 3(c). The reason is because the updating area includes the minimum gradient which cannot be handled appropriately by predictive watershed transform.

The proposed watershed transform examines the gradient values on the boundary between an updating block and an unchanged block. For example, A and B represent two pixels on the boundary of the two blocks (see Fig. 7(c)). Let P_A and P_B denote the gradient values of A and B, respectively. Region 5 has the root in the updating area in the previous frame, but that the root value is changed in the current frame. In this case, the two regions across the boundary are merged only when $P_A \leq P_B$ is true. On the other hand, the two regions cannot be merged when $P_A > P_B$. In this case, a watershed line is generated to segment the two regions. In Fig. 7(c), $P_A \leq P_B$ is true so that the two regions are merged. In the other case for the boundary between Block I and II, the boundary merge scheme proposed in Section III.2 is used because the root for the Region 1 is not included in the updating block.

4. Block-based skip of flooding operations

One advantage of a block-by-block operation is that a part of the watershed transform algorithm for a certain block can be skipped if the block meets some conditions for the skip. This advantage can also be applicable to the proposed block-based watershed transform. There exist two conditions that allow the skip of flooding operations. The first condition is that all gradient values in a block are identical. The second condition is that only a single local minimum exists in a block. If one of the two conditions is satisfied, the flooding operations are skipped. Fig. 8 shows the flowchart of the techniques for the flooding operation skip.

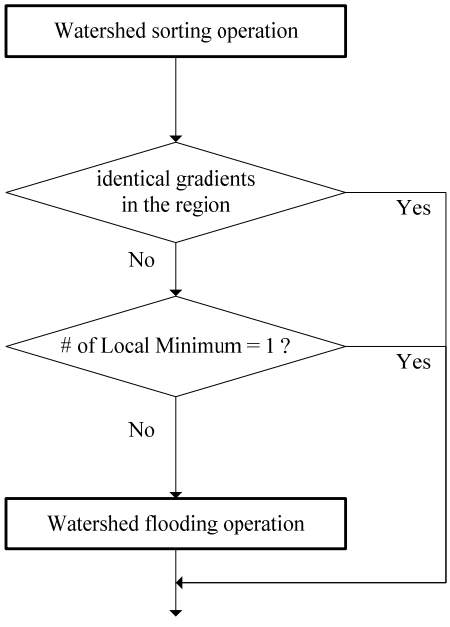


Fig. 8. Decision for skipping the flooding operation in a block.

5. The Algorithm

This subsection combines all the proposed techniques and presents the proposed watershed algorithm with the flow chart shown in Fig. 9. The proposed algorithm is composed of two main steps: sorting/flooding step and

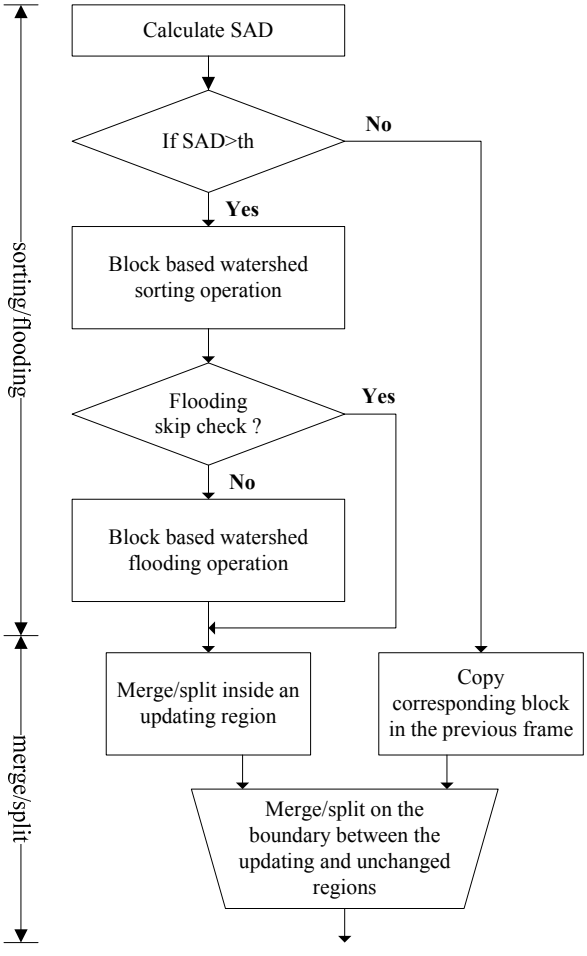


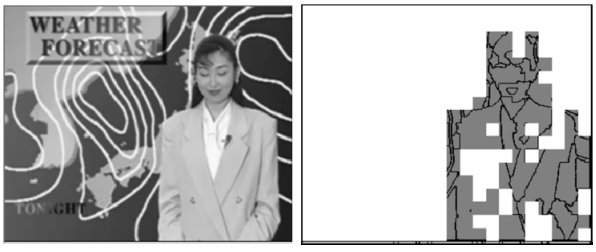
Fig. 9. Flow chart of the proposed watershed transform.

merge/split step. Each block is visited and its SAD is compared with a predefined threshold. The sorting operation is performed only when the SAD is greater than the threshold. Then, flooding skip is tested as explained in Section III.4. After the flooding operation is completed, the merge/split operation is performed inside the updating region first. Finally, the merge/split operation is performed on the boundary between the updating and unchanged regions.

IV. EXPERIMENTAL RESULTS

The proposed watershed algorithm is modeled in C-programming language and the simulation results are obtained with the model. The test videos are CIF-sized sequences and the input images are decomposed into 16x16 blocks for block-based operations. The sequences are tested using a single-core Intel i5-750 processor running at 2.67 GHz. Fig. 10(a) shows an example frame of Weather sequence and Fig. 10(b) shows the updating regions that are required to perform the watershed algorithm. In Fig. 10(b), the white blocks represent the regions that do not require an watershed update because the regions are same as those in the previous frame. The other blocks require the watershed transform to be performed to generate updated segmentation results. As shown in this figure, a large area is not required to be updated.

Table 2 shows the average percentage of the regions per a frame that are required to be updated in the proposed algorithm. Recall that the proposed algorithm updates a block when the SAD of the block is greater than 0 (see (2)). Note that the threshold for the update in the predictive watershed algorithm is 320. This implies that the updating area in the proposed algorithm is larger than that in the predictive watershed.



(a) (b)

Fig. 10. The region that requires updated watershed segmentation (a) 4th frame of Weather sequence, (b) Updating region indicated by shaded blocks.

Table 2. The percentage of the updating regions

	Container	Foreman	Hall monitor	Sean	Weather
percentage	47%	94%	41%	31%	30%

The percentage of the updating area depends on video sequences. Foreman sequence with a large and fast moving object requires a large updating area whereas other video sequences with less motion require less than a half of a frame to be updated. For these video sequences, a large amount of computation can be saved with the scheme that updates only the changed regions.

In order to evaluate the effect of updating regions, the original Vincent-Soille watershed transform, predictive watershed transform and the proposed algorithm are performed and the results are presented in Fig. 11. Fig. 11(a) shows the 20th frame of the Hall monitor sequence, and Fig. 11(b) shows the segmentation results by the original Vincent-Soille algorithm. Fig. 11(c) shows the

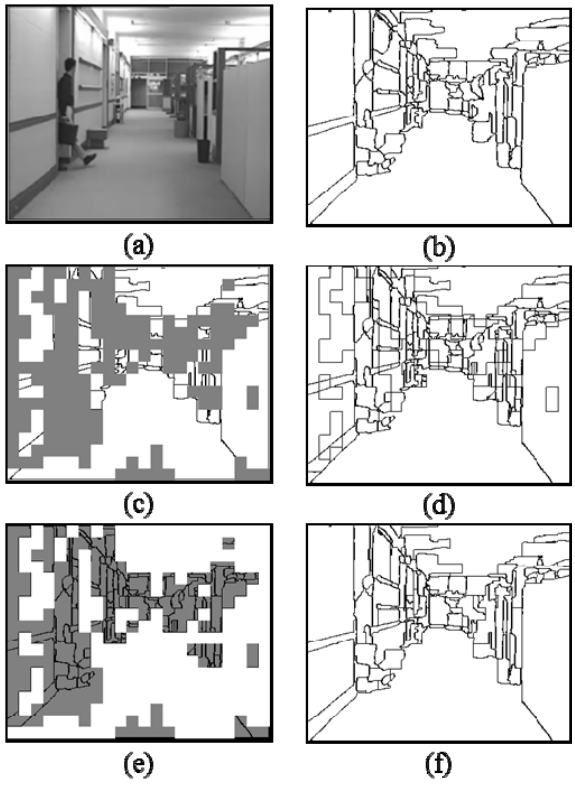


Fig. 11. Watershed segmentation results (a) Hall Monitor sequence, (b) Segmentation by Vincent-Soille algorithm, (c) Updating area, (d) Segmentation by predictive algorithm, (e) Segmentation inside the updating region, (f) Segmentation by the proposed algorithm.

updating area in the new frame, Fig. 11(d) shows the segmentation results by predictive watershed algorithm, Fig. 11(e) shows the segmentation results inside the updating region generated by the proposed algorithm described in Section III.B, and Fig. 11(f) shows the segmentation results by the proposed watershed algorithm. Comparison between the results in Fig. 11(b), (d) and (f), shows that the segmentation results of the proposed algorithm are very close to those of the original Vincent-Soille algorithm whereas there exist mismatches between the segments in Fig. (b) and (d).

The accuracy of the proposed watershed algorithm is compared with that of predictive watershed algorithm. The numbers of segments generated by the original Vincent-Soille, predictive watershed, and the proposed algorithms are compared. Fig. 12 shows the numbers of segments for 50 frames of Weather video sequence. As shown in the figure, the number obtained from predictive watershed is quite different from that from the original Vincent-Soille algorithm. However, the proposed algorithm generates almost the same number as the original algorithm does.

The numbers of segments are also compared with additional four video sequences: Container, Foreman, Hall monitor, Sean. The average number of segments per a frame is presented in Table 3. The second row gives the average number of segments per a frame generated by the original Vincent-Soille algorithm. The third and the fourth rows present the RNDRs (Region Number Difference Ratios) for predictive and the proposed watershed algorithms, respectively. As shown in this table, the proposed watershed algorithm generates much accurate results than predictive watershed algorithm for all 5 video sequences.

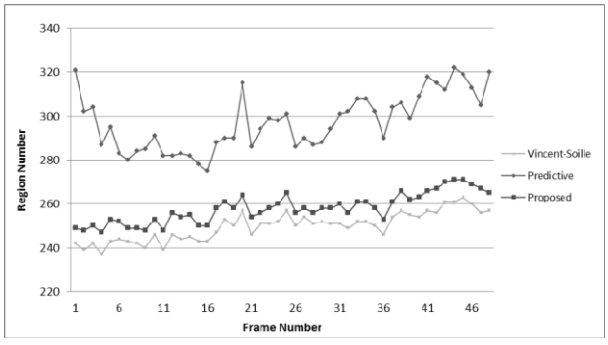


Fig. 12. The number of segments in 50 frames of Weather sequence.

Table 3. The average number of segments in 5 video sequences

	(1)	(2)	(3)	(4)	(5)
Average Number of segments generated by Vincent-Soille algorithm	125	93	117	119	250
Predictive	57.5%	14.6%	91.9%	33.9%	18.6%
Proposed	9.6%	3.8%	15.3%	2.7%	3.1%

(1) : Container (2) : Forman (3) : Hall monitor
(4) : Sean (5) : Weather

Table 4. Execution time comparison of Vincent-Soille algorithm, predictive and proposed algorithms

	Container	Sean	Weather
Vincent-Soille's algorithm	50 ms	54 ms	67 ms
Predictive watershed	25.6 ms	18.9 ms	22.2 ms
proposed	26.8 ms	19.9 ms	21.8 ms

Table 4 shows the execution time of the Vincent-Soille, the predictive and the proposed algorithms. The execution time is averaged over 50 frames. Only a single processor is used to measure the execution time. For fair comparison, the thresholds for both the predictive watershed and the proposed watershed choose the same value of 0. The measured times only include the computation time, but do not include the period for data loading, distribution, coalescence or saving.

As shown in the table, both the predictive and the proposed watershed algorithms require much less execution time than the original Vincent-Soille algorithm. This is because the amount of the computation is reduced by skipping operations required for the unchanged regions. The computation times for the predictive watershed and the proposed watershed algorithms are about the same. The proposed watershed requires additional computation to handle the merge/split operations on the block boundaries. However, the amount of computation is reduced by the flooding skip presented in Section III.4.

The above experimental results show the advantage of the proposed algorithm over the predictive watershed transform because the accuracy of the proposed algorithm is better than the predictive watershed transform (see Table 3) while both algorithms require about the same amount of computation. An additional advantage of the proposed algorithm comes from the fact that the block-based operation makes it easy to

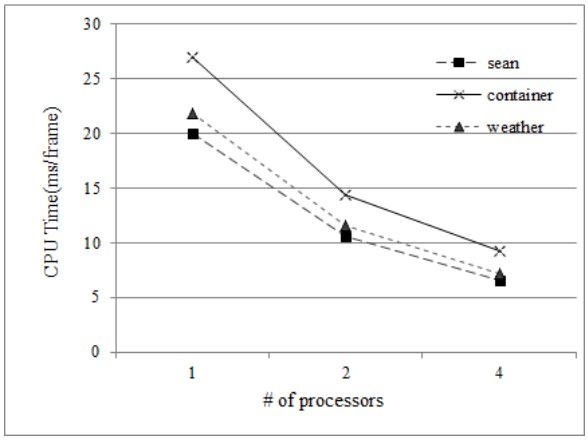
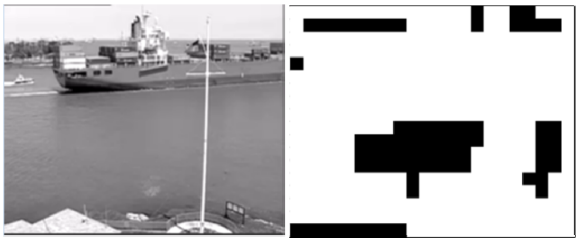


Fig. 13. Computation time versus the number of processors.



(a) (b)

Fig. 14. The blocks for which flooding operations are skipped; (a) An example frame in Container sequence, (b) Black blocks represent the skipped blocks.

parallelize the algorithm. In order to estimate the speed-up for parallel execution, additional simulations are performed. Fig. 13 shows the estimated execution time when the proposed algorithm runs on 2 and 4 processors. Note that parallelization requires additional computation to handle the boundaries of the blocks assigned to different processors and the results shown in Fig. 13 account for this additional computation. For 2 processors, the speed-up is about 1.9 whereas, for 4 processors, the speed-up is greater than 3.

Fig. 14 shows the regions that can skip the flooding operations as proposed in Section III.4. Fig. 14(a) shows an example frame of Container sequence and Fig. 14(b) shows the black blocks for which the flooding operations are skipped. The average percentages of the skipped blocks are obtained with five video sequences and the results are given in Table 5. Between 6% and 21% of blocks are skipped by the technique proposed in Section III.4. Note that this percentage is obtained from the entire image.

Table 6 shows the ratio of the execution time required by flooding operations over the total execution time of the watershed transform. For all 5 video sequences, about 90% of computation is performed for flooding operations. Note that this percentage is obtained from the execution time of an entire image.

Table 5. Percentage of the blocks for which flooding operations are skipped

	Container	Foreman	Hall monitor	Sean	Weather
Flooding skip block	21%	6%	10%	7%	10%

Table 6. Execution time of the flooding skip effect

	Container	Foreman	Hall monitor	Sean	Weather
Flooding operation ratio	89.5%	90.5%	89.9%	90.5%	92%

V. CONCLUSIONS

Experimental results indicate that the proposed algorithm provides almost the same segmentation accuracy as the Vincent-Soille watershed transform whereas the previous work, predictive watershed transform, results in substantially different segmentations. Since the proposed block-based watershed method allows multiple blocks to be processed independently of different blocks, the algorithm can be implemented in hardware without much difficulty for handling the complex control for global operations in the watershed transform.

ACKNOWLEDGMENTS

This work was supported by Industrial Strategic Technology Development Program funded by the Ministry of Knowledge Economy (MKE, Korea) (10039188, Development of multimedia convergence programmable platform for smart vehicles)

REFERENCES

[1] J.B.T.M. Roerdink and A. Meijster, "The Watershed Transform: Definitions, Algorithms and Parallelization Strategies," *Fundamenta Informaticae*, Vol.41, nos. 1-2, pp.187-228, 2001.

[2] R. Gonzalez and R. Woods, *Digital Image Processing*. Upper Saddle River, NJ: Prentice-Hall, 2007.

[3] Y. Tsai, C. Lai, Y. Hung, and Z.Shih, "A Bayesian approach to video object segmentation via merging 3-D watershed volumes," *IEEE Trans. on Circuits and Systems for Video Technology,* Vol.15, No.1, pp.175- 180, Jan., 2005.

[4] H. Xu, A. A. Younis, and M. R. Kabuka, "Automatic moving object extraction for content-based applications," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol.14, No.6, pp.796-812, Jun., 2004.

[5] S. Chien, Y. Huang, B. Hsieh, S. Ma, and L. Chen, "Fast video segmentation algorithm with shadow cancellation, global motion compensation, and adaptive threshold techniques," *IEEE Trans. on Multimedia,* Vol.6, No.5, pp.732-748, Oct., 2004.

[6] A.N. Moga and M. Gabbouj, "Parallel image component labelling with watershed transformation ," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.19, No.5, pp.441-450, May, 1997.

[7] A. Moga, B. Cramariuc, and M. Gabbouj, "An Efficient Watershed Segmentation Algorithm Suitable for Parallel Implementation," in *Proc. of IEEE Int'l Conf. Image Processing,* Vol.2, No., pp.101-104 Oct., 1995.

[8] S.-Y. Chien, Y.-W. Huang, and L.-G. Chen, "Predictive watershed: a fast watershed algorithm for video segmentation," *IEEE Trans. on Circuits and Systems for Video Technology,* Vol.13, No.5, pp.453-461, May, 2003.

[9] B. J. Mealy, "Scanning Order Dependencies in Watershed Transform", *Technical Report UCSC-CRL-02-37*, Dec., 2002.

[10] L. Vincent, and P. Soille, "Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol.13, No.6, pp.583-598, Jun., 1991.

[11] J. S. Kim, H. J. Lee, T. H. Lee, M. J. Cho, and J. B. Lee, "Hardware/Software Partitioned Implementation of Real-time Object-oriented Camera for Arbitrary-shaped MPEG-4 Contents," in *Proc. Of IEEE/ ACM/IFIP Workshop on Embedded Systems for Real Time Multimedia,* pp.7-12, Oct., 2006.

[12] J. S. Kim, J. H Zhu, and H. J. Lee, "Block-Level Processing of a Video Object Segmentation Algorithm for Real-Time Systems," in *Proc. Of IEEE International Conference on Multimedia and Expo,* pp.2066-2069, 2-5 Jul., 2007.

[13] N. Hirai, T. Song, Y. Liu, and T. Shimamoto, "A Novel Spiral-Type Motion Estimation Architecture for H.264/AVC," *Journal of Semiconductor Technology and Science*, Vol.10, No.1, pp.37-44, Mar., 2010.

Jung-Whan Jang received the B.S. degree in electrical engineering from Korea Advanced Institute of Science and Technology(KAIST), Daejeon, Korea, in 1998 and the M.S. degree in electrical engineering from Yonsei university, Seoul, Korea, in 2000. He is working toward Ph.D degree in electrical engineering of Seoul National University. From 2000 to 2007, he worked at the Video Security System Development Group of Samsung Electronics Ltd., Suwon, Korea, as a Senior Engineer. His major research interests are in the area of image segmentation and the algorithm and architecture of H.264/AVC.

Hyuk-Jae Lee received the B.S. and M.S. degrees in Electronics Engineering from Seoul National University, Korea, in 1987 and 1989, respectively, and the Ph.D. degree in Electrical and Computer Engineering from Purdue University at West Lafayette, Indiana, in 1996. From 1998 to 2001, he worked at the Sever and Workstation Chipset Division of Intel Corporation in Hillsboro, Oregon as a senior component design engineer. From 1996 to 1998, he was on the faculty of the Department of Computer Science of Louisiana Tech University at Ruston, Louisiana. In 2001, he joined the School of Electrical Engineering and Computer Science at Seoul National University, Korea, where he is currently working as a Professor. He is a founder of Mamurian Design, Inc., a fabless SoC design house for mobile multimedia applications. His research interests are in the areas of computer architecture and SoC design for multimedia applications.