J Sign Process Syst (2008) 52:281–295
DOI 10.1007/s11265-007-0157-3

An Efficient Hillclimbing-based Watershed Algorithm and its Prototype Hardware Architecture

C. Rambabu · I. Chakrabarti

Received: 13 July 2007 / Revised: 11 November 2007 / Accepted: 14 November 2007 / Published online: 10 January 2008

Abstract Image segmentation is the process of isolating objects in an input image, that is, partitioning the image into disjoint regions, such that each region is homogeneous with respect to some property, such as gray value or texture. Watershed-based image segmentation has gained much popularity in the field of biomedical image processing and computer vision where large images are not uncommon. Time-critical applications like road traffic monitoring, and steel fissure analysis require fast realization of the segmentation results. The present paper proposes a fast watershed transform based on hillclimbing technique. The complexity of the algorithm has been reduced by doing away with multiplication normally required to form a lower complete image in an intermediate step of the overall segmentation. The reduced complexity makes the algorithm suitable for dedicated hardware implementation. An FPGA-based architecture has been developed to implement the proposed algorithm involving moderate hardware complexity. This architecture enhances the applicability of this algorithm for real-time applications.

Keywords image segmentation · watershed transformation · hillclimbing technique · FGPA implementation

C. Rambabu (✉)
Imaging Informatics Group, Bioinformatics Institute,
Singapore 138671, Singapore
e-mail: chinta@bii.a-star.edu.sg

I. Chakrabarti
Electronics and Electrical Communication Engineering
Department, Indian Institute of Technology,
Kharagpur 721 302, India

1 Introduction

Image segmentation is a preliminary step to several image analysis tasks like object recognition, object-based image compression and context-based indexing of images. Segmentation of an image involves partitioning it into a number of homogeneous segments. Otherwise, one can view segmentation as a pixel labelling process in which the same label is given to all pixels that belong to a homogeneous region. A collection of techniques [1–3] have been suggested as possible solution to the inherently difficult image segmentation problem. They fall into one of the following categories, namely histogram-based algorithms [4–6], edge-based algorithms [7–9], region-based algorithms [10–15], Markov Random Field-based algorithms [16–18] and clustering-based methods [19–21]. The watershed transform method, which belongs to the broad class of region-based segmentation approach, is simple to formulate and it can effectively recognize the important closed contours of a given image. Watershed transformation methods have found numerous applications such as industrial, biomedical, and computer vision applications. As concrete examples, one can mention here the application of watersheds to automated analysis of image acquired during oil exploration, road traffic analysis, and to segmentation of several types of images including electrophoresis gels, overlapping grains, fissure facets in a steel fracture, and 3-D holographic images. Watersheds have also been used in extracting and tracking objects in time sequences of a moving heart in nuclear medicine, in counting objects in noisy, anisotropic 3-D biological images, and for segmenting the internal structures in 3-D MR images of brain.

A brief idea of the watershed construction is given as follows. A gray scale image is considered as a topographic

relief, the gray scale value of a pixel being the altitude at that particular point. Now, let a drop of water fall on such a topographical surface. By gravity, it will flow down along the steepest slope path until it reaches a point or region of minimum. The whole set of points of the surface, whose steepest slope path reaches a given minimum, constitutes the *catchment basin* associated with this minimum. Each pair of adjacent catchment basins are separated by watershed lines. Thus, raindrops falling on both sides of a watershed line flows into different catchment basins. The watershed transform has been approached from two different perspectives. One class of algorithms aims to detect the catchment basins by simulating the flooding process while the second tracks directly the watershed lines by using the arrowing techniques. More efforts though have been concentrated on the first type of algorithm. Among these, there are algorithms which construct the watershed lines and the algorithms which do not delimit the basins by watersheds (0-width watershed lines).

The fast implementation of immersion-based watershed algorithm has been introduced by Vincent and Soille [22]. The authors simulated a flooding process, in which the water is coming up out of the ground and flooding the catchment basins without predetermining the regional minima. Utilizing a First-In-First-Out (FIFO) structure, the pixels at altitude $h + 1$ are processed after those at altitude h. Due to the processing of pixels at altitude h in every iteration, the problem is reduced to calculating the *geodesic skeleton of influence zones* (SKIZ). Whenever water bodies originating from different catchment basins reach each other, a dam is built to prevent the basins from merging. This approach is implemented in software by several data structures, including graphs and grids with an arbitrary connectivity. Alternatively, an ordered queue based watershed algorithm has been proposed by Meyer [12]. The common strategy described first determines the regional minima independent of their altitude. Afterward, the adjacent pixels of these minima are added to a hierarchical queue. At each iteration, the pixels with the lowest altitudes are popped from the queue and then processed. This step is repeated until all pixels are processed.

Finally, several shortest-path algorithms for the watershed transformation with respect to topographical distance can be found in literature [23, 24]. In fact, these methods compute the shortest paths between the regional minima and all other pixels. Here, a new definition of watershed and catchment basins in terms of a distance function and topographic distance has been introduced [24, 25]. Thus, a pixel is incorporated in the catchment basin associated with the "closest" regional minimum. Consequently, each pixel has an ancestor according to the flooding order, which is a neighbor with the precedent distance value. These shortest paths are tracked by walking upward (hillclimbing) or downward (rainfalling) on the topographic surface according to the precedence relation imposed by distance. In the rainfalling simulation, a non-minimum pixel is labelled on a path along which a drop of water would slide towards a minimum, when it starts falling from that non-minimum point in the topographic surface.

On the contrary, in the hillclimbing simulation, regions grow independently around their regional minima. The labels of minima are propagated upward to the higher neighboring pixels which do not have neighbors along a steepest path other than the given candidate for region growing. A hillclimbing-based watershed algorithm has been introduced by Meyer [24]. In this method, the regional minima are first computed and then the adjacent pixels of these minima are added to a hierarchical queue. At each iteration, the pixels with the lowest altitude are dequeued from the queue and then one labels all the upper neighbors of the current pixel along the steepest slope, unless the neighboring pixel is already labelled and the label differs from the current pixel label, in which case the neighboring pixel is classified as a watershed pixel. This step is repeated until all the pixels are processed, thus simulating an overflooding of the processed data (called *hillclimbing*). These algorithms provide a straightforward way to find the watershed lines but require extensive computations. In general, the immersion-based methods of the type discussed earlier are much faster than the shortest path-based algorithms.

Meyer [12] proposed an algorithm based on an ordered queue, which in fact is made of 256 queues, one for each of the possible grey levels. Recently, an improved technique of simulated flooding based watershed algorithm [26] has been proposed. Involving only one single queue, it exhibits equivalent performance at reduced hardware complexity while compared to Meyer's algorithm [12]. However, the flooding based algorithm [26] suffers from an important drawback in that no synchronization in label propagation is possible in a non-minimum plateau which has more than one closely located regional minima. It leads to a consequent loss in performance. The present paper describes a hillclimbing based technique which overcomes the drawback of the flooding-based algorithm [26]. The proposed hillclimbing technique is also faster than the flooding method [26]. For, introducing a steepest lower complete image in the hillclimbing algorithm reduces the number of searching points, whereas the label propagation in the flooding-based method [26] involves a depth-first search mechanism, processing one local minimum at time. Moreover, the proposed method has a reduced time complexity as it does away with inherently time-consuming lower complete transformation normally involved in a conventional hillclimbing method [14, 24]. The conventional hillclimbing technique [14] which works well with

regular gray-scale images, is based on lower complete transformation that involves time-consuming multiplication by a constant of the gradient image based on geodesic distance. The reduced complexity also paves the way for an economical hardware implementation of the proposed algorithm. A dedicated hardware architecture for implementing the watershed segmentation algorithm in needed for time-critical application like traffic monitoring, video surveillance, etc. A prototype architecture for implementing the proposed algorithm is also given in the present paper. The rest of the paper is organized as follows.

The next section gives a detailed discussion of the conventional and proposed hillclimbing technique and then analyzes the computational complexity of the principal steps of the proposed algorithm. Section 4 presents the simulation results of running the proposed algorithm and the conventional algorithm on different test images for comparison. In addition, this section evaluates the quality of segmentation produced by the proposed algorithm in a quantitative manner. An FPGA-based prototype architecture has been developed to implement the proposed algorithm involving moderate hardware complexity, and its implementation results are presented in section 5. Finally, section 6 concludes this paper.

2 Hillclimbing Simulation

The sequential watershed transform method based on hillclimbing simulation has been proposed by Meyer [14, 24]. The conventional algorithm [14, 24] starts by detecting and labelling the initial seeds, that is, the minima of the gradient image, which characterize the regions of interest in a given image. The latter are defined as connected plateaus of pixels in the gradient image which do not have neighboring pixels of lower gray level (plateau of minima). Starting from the minima, region growing is then performed. Thus, the non-labelled pixels are assimilated into different components in an increasing order of gray levels. A characteristic of this flooding is that waves always progress upward, or are synchronized inside a plateau. Inside the flat areas, which are not yet labelled and are called plateaus of non-minima, components progress synchronously, such that they incorporate equal extents within the plateau. Consequently, a pixel on such a plateau is labelled along the shortest path completely included in the plateau to a lower downward brim of the plateau. During the flooding of a topographic surface, there appears a dual relation between the pixels, as stated below.

Property 1 A pixel p acquires a label from q iff p has a higher gray value than q, or p and q have the same gray value, but q is closer to a downward brim of its plateau than p.

The measure of closeness of a pixel within a plateau to a lower border is the geodesic distance called the *lower distance* which is defined as follows.

Definition 1 (Lower distance) The lower distance d of a pixel p is as follows: $d(p)$=0 if pixel p is a minimum; otherwise, $d(p)$ is equal to the length s of the shortest path between two pixels p and q such that $\forall i \in \{1, 2, \ldots, s\}$, the line $(p_{i-1}, p_i) \in G$ (G being the underlying rectangular grid), $p_0 = p$, $p_s = q$ and $F(q) < F(p)$. Moreover, if $s > 1$, $\forall i \in \{1, 2, \ldots, s-1\}$, $F(p_i) = F(p_{i-1})$. $F(.)$ represents the gray scale image.

A sequential watershed transform, which does not construct watershed lines, has the drawback of being scanning order dependent and hence, of producing inaccurate results in some cases. To overcome this problem, Meyer has introduced a lower-complete transform computation [23]. From two images, namely the gray scale image F and the lower distance image d (vide definition 1), a lower-complete image can be defined to include the ordering relations imposed by gray level and lower distance. This image is known as *lower-complete* [14] and it is defined as follows.

Definition 2 (Lower-Complete Transform) The mapping $l(p) = \lambda * F(p) + d(p), \forall p \in D_F$, is called the lower-complete transformation of an image F based on the lower distance image d. Note that $\lambda > d(p), \forall p \in D_F$.

Some representative values for λ are $max_{p \in D_F}\{d(p)\} + 1, max_{0 \le h \le H}\{\text{Histogram}[F(h)]\} + 1$, where *Histogram* is an array used to store the number of pixels with the gray level h for each gray level $h \in \{0, 1, \ldots, H\}$ in the image F. Another possible value of λ is $N \times M + 1$, where $N \times M$ is the size of the image F. Note that in a lower-complete image l, every pixel $l(p)$ has a lower neighbor, except for the minima. It is evident that on a lower-complete image, any pixel p can possibly have one or more neighbors with lower values, of which the one with the minimum value may be referred to as the *steepest lower-complete* (*slc*) neighbor of p, and denoted as $slc(p)$. It is not however, necessarily unique. Flooding process is applied to a lower-complete image [23], in which the shortest paths actually correspond to the lines of the steepest slope. Moreover, every non-minima pixel is reached by flooding from its steepest lower-complete neighbor. Consequently, flooding should progress only between two neighboring pixels whose lower-complete transformed values satisfy the ordering relation involved in propagation of labels. Once a pixel has received a label, it is incorporated in the region it belongs to. In this algorithm, regions grow independently around their seeds, namely, the regional minima. More

specifically, labels of minima are propagated from a candidate pixel upward to the higher neighboring pixels which do not have any neighbor with value lower than that of the given candidate for region growing. This bottom-up method is known as *hillclimbing simulation*.

In the conventional algorithm [14], a raster scan is used to perform the flooding process, in which a FIFO queue is initialized with the pixels within plateaus of regional minima. These pixels have non-minima pixels in their neighborhood. An additional image *slc* is used to store the altitudes of the steepest lower complete neighbors of the corresponding pixels. A candidate pixel p removed from the queue propagates its label to all its neighboring pixels q if there is an *arc* from p to q in the forest, that is, $slc(q) = l(p)$. Any new labelled pixel becomes a candidate, and it is inserted in the FIFO queue. When the queue of candidates is emptied, each pixel in the output image gets labelled and appended to a single connected component and the hillclimbing procedure stops. The set of non-overlapping components contributes a complete partition of the image.

3 Proposed Hillclimbing Simulation

In Moga's algorithm [14], first the input gradient image has been transformed into a lower-complete image. This transformation involves $O(m \times n)$ multiplications and additions, where $m \times n$ is the size of the input image. The proposed algorithm attempts to reduce the computational complexity in two ways. First, the lower complete image computation is avoided. Secondly, the plateau detection and its labeling is computed in one scan rather than two scans by the use of two queues. Flooding of the non-minima plateau pixels has been done by directly using the distance image. Also in Moga's algorithm, the raster scan operation has been employed twice, first in "Minima Detection and Labeling" process and next in "Simulated Flooding" process. During the second raster scan, minima and plateaus of minima, which have been labeled during the first raster scan, are stored in a FIFO queue. In the proposed algorithm, however, starting from the minima, the recursive label propagation (flooding) is performed in breadth-first order using a single FIFO queue, a distance image and a steepest lower neighbor image *(sln)* which stores the steepest lower neighboring altitude. This steepest lower image is computed during the first raster scan (local minima detection and labeling process) rather than the following scan. Thus the neighboring pixels need not be accessed again. This improves the complexity of the algorithm. Moreover, during plateau analysis, the outer pixels are stored in an array *Outers*, which will be used in the procedure of computing the lower distance for subsequent label assignment. Amongst the pixels belonging to a plateau, it may be recalled that those having at least one neighbor with lower altitude are termed *outer* pixels while the others are called *inner* pixels. In Moga's algorithm, these outer pixels are first inserted in a FIFO queue, and then the lower distance computation has been carried out for inner plateau pixels. Time required for movement of pixels from *Outers* array to the FIFO queue can be diminished by employing another FIFO queue, where the outer pixels will be directly enqueued. The detailed description of the two major phases of the proposed hillclimbing technique is provided next.

3.1 Detection and Labeling of Local Minima

In this procedure, a single raster scan has been employed, and two FIFO queues Q_{PD} (for plateau detection) and Q_{PA} (for plateau analysis) have been utilized as shown in Fig. 1.

For each not yet labelled pixel p, its 4 or 8-connected neighborhood is inspected. Thus if all the neighbors are of higher gray level than p, then p is deemed to be an *isolated minimum*, and a label is assigned to it. Else, if this pixel belongs to a plateau, the plateau is scanned in a breadth-first order, and the visited pixels are labelled with the current label, and inserted into the queue Q_{PD} for detection of plateau. The examination always starts from an *inner* pixel which introduces the neighboring pixels of equal altitude in the list of candidates. A currently investigated candidate pixel may still qualify as an *inner* pixel; otherwise, it is an *outer* pixel. The lower distance of the outer pixel is 1, and inserted into queue Q_{PA} for analysis of plateau. After scanning the plateau, if the queue Q_{PA} is not empty (indicating that a non-minima plateau is detected), it is then necessary to compute the distance of *inner* pixels (from the outer pixels) and assign NARM (not a regional minimum) label to the plateau currently detected for subsequent flooding. Also the steepest lower neighboring altitude pixels are computed and stored in an additional image known as the steepest lower neighbor image.

Definition 3 (Steepest lower neighbor image) The steepest lower neighbor image, denoted by *sln*, corresponding to a gray scale image F, is derived from the latter as follows.

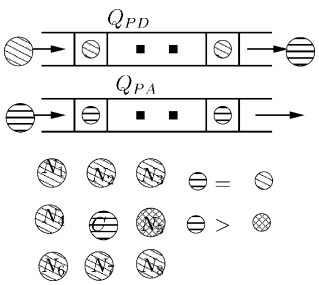

Figure 1 Local minima detection and labeling process.

Each pixel $p \in D_F$ is replaced by its steepest lower neighbor, given by $sln(p) = min_{q \in N_G(p)}\{F(q)\,|F(q) < F(p)\}$, if p is an outer pixel; otherwise, $sln(p) = F(p)$.

The pseudo-code of the whole procedure of detection and labelling of local Minima is given below. Procedures Neighborhood_Inquiry and Plateau_Analysis, described next in the form of a pseudocode as Procedure 2 and 3 respectively, involve detection and labeling of plateaus.

3.2 Flooding of Non-Minima

Another raster scan procedure is used to perform the flooding process, in which a FIFO queue is initialized with the pixels within plateaus of minima, as discussed in the previous section. Consequently, flooding should progress only between two neighboring pixels whose steepest lower complete neighbor and distance values satisfy the flooding ordering relation. Once a pixel has received a label, it is correctly incorporated in the region it belongs to. The basis of the proposed flooding order is established by the following theorem.

Theorem 1 $p \succ q$ (to be read p is flooded from q) in the grayscale image F whose lower distance image and steepest lower neighbor image are d and sln respectively if

$$F(q) = F(r) \mid \{r \in N_G(p)\} \wedge \{sln(p) = F(r)\} \quad (1)$$

else, $p \succ q$ if

Procedure 1 Local Minima Detection & Labeling(F, L, d, sln)

```
1: Input: Morphological multi-scale gradient image F. {F : D_F → ℕ}
2: Output: Label image L, distance image d and steepest lower image sln. {L, d,
   sln : D_F → ℕ }
3: Let G be a subset of ℤ² × ℤ² and N_G(p) stands for the neighbors of a pixel p
   on the grid G.
4: INIT ← −2;{/* initialization value */}
5: NARM ← −1;{/* Not A Regional Minima */}
6: MAX_DIST ← −1;
7: current_label ← 0 {/* initialize the Current label */ }
8: for all (p ∈ D_F) do
9:    L(p) ← INIT; d(p) ← MAX_DIST; /* label image and distance image ini-
      tialization */
10: end for
11: for all (p ∈ D_F with L(p) = INIT)(Raster scan) {/* for each pixel in the image
    which is not yet visited */} do
12:    Neighborhood_Inquiry(p, F, pixel_type);    /* procedure for inquiring the
       neighborhood pixels of p */
13:    if pixel_type = MINIMUM {/* It is a isolated minimum */} then
14:       L(p) ← current_label++;
15:    else if pixel_type = ON_PLATEAU {/* A plateau is detected */} then
16:       Plateau_Analysis(p, F, L, d, current_label);    /* procedure for plateau
          detection and its labeling */
17:    else if pixel_type = NON_MINIMUM {/* a non-minimum pixel is detected
       */} then
18:       L(p) ← NARM;
19:    end if
20: end for
21: /* End of procedure LM_Det_Labeling */
```

Procedure 2 Neighborhood_Inquiry(p, F, $pixel\_type$)

```
1: Input: p, F, pixel_type.
2: Output: pixel_type.
3: pixel_type ← MINIMUM;
4: for all q ∈ N_G(p) {/* N_G(q): neighbors of p */} do
5:    if F(q) = F(p) {/* plateau detection */} then
6:       pixel_type ← ON_PLATEAU;
7:    else if F(q) < F(p) {/* non-minima detection */} then
8:       pixel_type ← NON_MINIMUM;
9:    end if
10: end for
11: /* End of procedure Neighborhood_Inquiry */
```

$$\{F(p) = F(q)\} \wedge \{sln(p) = F(q)\} \wedge \{d(p) = d(q) + 1\} \quad (2)$$

Proof (Proof of Eq. (1)) Consider a pixel $q^{\sim} \in N_G(p)$ such that q satisfies the condition 1. Flooding always proceeds in increasing order of gray levels. Thus, considering flooding at level $F(q)$, it is either true that p is not yet flooded and it will be flooded by q; else, it has been already flooded from another neighbor $r^{\sim} \in N_G(p)^{\sim}(r \neq q)$. If the second case holds, then the gray levels higher than the gray level $F(q)$ will be flooded only after the gray level $F(q)$ has been flooded. Thus $F(r) \leq F(q)$. However, as q satisfies (Eq. (1)), it happens to be the neighbor of p with the lowest surrounding gray level. Thus, $sln(p) = F(q) = F(r)$. Hence, pixel p is flooded from q or r, depending on which one is scanned first. however, flooding always occurs from the surrounding neighbor with the lowest gray level, although this neighbors may not be unique.

(Proof of Eq. (2)) The lower distance (vide definition 1) is a measure of closeness of a pixel within a plateau to its lower brim. It also represents the time taken by the wave which advances from the closest lower brim of the plateau and floods the pixel. Thus, if pixel p on a plateau has its lower distance $d(p) > 1$, it will be flooded from a pixel which has been flooded by a wave at time $d(p) - 1$.

To prove (Eq.(2)) by contradiction, suppose p inherits its label from another neighbor r with lower distance $d(r) > d(p) - 1$. It means that the wave from r reaches the pixel p at time $d(r) + 1$ to find p without a label. From the hypothesis, $d(r) + 1 > (d(p) - 1) + 1$ or $d(r) + 1 > d(p)$. That is, two wavefronts which reach p along paths of shortest length fail to immerse the plateau at same rate. Hence, (Eq. (2)) is established by contradiction.

In this procedure, only one FIFO queue data structure Q_F has been used for recursive label propagation of the candidate pixels. First, one stores the seed pixels in the queue Q_F; next, labelling starts as follows. A candidate pixel p is removed from the queue Q_F, and its label is assigned to each of its neighboring NARM pixel q, which has a higher altitude. If the neighboring pixel q has the same altitude as p, then the distance image needs to be made use of to determine the label of q. Only those neighboring pixels, whose lower distance is one greater

Procedure 3 Plateau_Analysis(p, F, L, d, *current_label*)

```
Input: p, F, L, d, current_label.
Output: L, sl, current_label.
L(p) ← current_label;
FIFO_INIT(Q_PD); FIFO_INIT(Q_PA);{/* initialize the FIFO Queues (Q_PD) and (Q_PA)*/}
FIFO_ADD(p, Q_PD); {/* insert the pixel p into the queue Q_PD */}
while fifo queue Q_PD not empty {/* plateau Detection phase */} do
   p ← FIFO_REMOVE(Q_PD); {/* get candidate pixel p from queue Q_PD */}
   min ← F(p);
   for all q ∈ N_G(p) {/* inquire neighboring pixels of p */} do
      if ((L(q) = INIT) and (F(p) = F(q))) then
         L(q) ← current_label;
         FIFO_ADD(q, Q_PD);
      else if ((F(q) < F(p)) and (d(p) = MAX_DIST)) {/* outer pixel detection */} then
         d(p) ← 1;
         FIFO_ADD(q, Q_PA);
      end if
      if (F(q) < F(p)) then
         min ← F(q);
      end if
   end for
   sln(p) ← min; {/* collect steepest lower neighbor value into the sln image */}
end while
if FIFO QUEUE Q_PA is empty {/* a minima plateau is detected */} then
   current_label := current_label + 1;    /* increment the current_label value */
else
   /* a non-minima plateau is detected */
   Compute_Lower_Distance(L, d, Q_PA);    /* procedure for computing the lower distances of the inner pixels */
end if
/* End of procedure Plateau_Analysis */
```

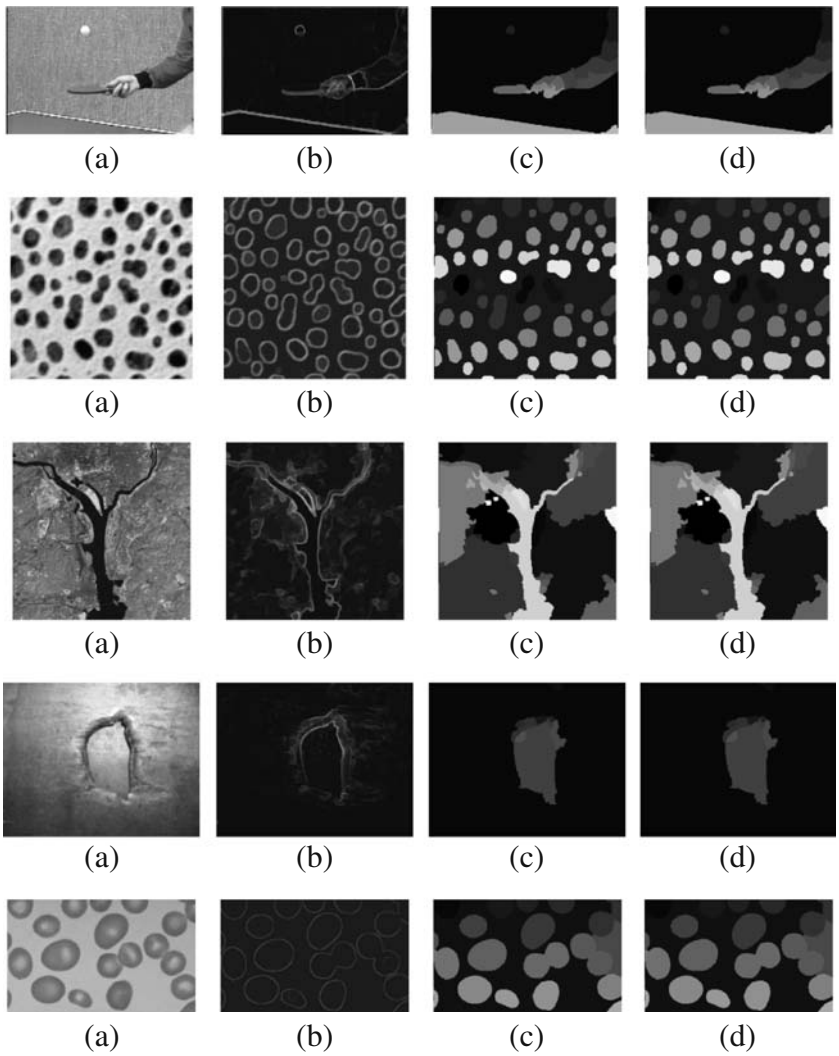

Figure 2 (a). Original image. (b). Morphological multi-scale gradient image with a threshold value of 13. (c). Segmentation produced by Moga's watershed algorithm [14]. (d). Segmentation produced by the Proposed watershed algorithm.

than that of the candidate pixel, get labelled. Any new labelled pixel becomes a candidate, and it is inserted in queue Q_F. This process is then repeated until the queue becomes empty. The pseudo-code of the flooding process is given below.

3.3 Complexity Analysis

Let $n = M \times N$ be the dimension of the image F whose domain is denoted as $D_F \subset \mathbb{Z}^2$, and G is a subset of $\mathbb{Z}^2 \times \mathbb{Z}^2$. Let N_G stand for the neighborhood pixels on the grid G.

3.3.1 Minima detection and its labeling

Let $p_1, p_2, p_3, \ldots, p_P$ be P plateaus, each p_i having n_i pixels and P_N be the detected non-minima plateaus.

1. The number of comparisons for plateau detection and plateau analysis is $\sum_{i=1}^{P} n_i * N_G$.
2. The number of comparisons for lower distance computation and non-minima plateau labeling is $\sum_{i=1}^{P_N} n_i * N_G$.

Total number of comparisons for minima detection and its labelling is then

$$LMD = (\sum_{i=1}^{P} n_i + \sum_{i=1}^{P_N} n_i) * N_G \approx n \tag{3}$$

3.3.2 Flooding process

Let $l_1, l_2, l_3, \ldots, l_L$ be the L local minima plateaus in the image G; moreover, let l_j have m_j pixels which have at least one NARM neighborhood pixel. The total number of

Table 1 Comparison between simulation time (in seconds) of Moga's hillclimbing technique [14] and the proposed technique.

Test Image	h	No. of regions	Time (*Sec*)	
			Moga's [14] Method	Proposed method
Claire (352×288)	10	17	0.092	0.075
Tennis (352×288)	13	22	0.095	0.072
Heart (256×256)	8	14	0.083	0.065
Akiyo (164×144)	14	22	0.031	0.024
MRI Brain (256×256)	12	38	0.072	0.055
Cermet (256 × 256)	30	61	0.081	0.064
Washinton (250×250)	16	46	0.098	0.073
Steel Fissure (290×369)	15	10	0.105	0.091
Blood Cells (212×353)	26	20	0.078	0.065

comparisons for the entire label propagation (in the worst case) is

$$LP = n + (\frac{1}{P_N}\sum_{i=1}^{P_N} n_i) * (\sum_{j=1}^{L} m_j * N_G) \simeq 2n \quad (4)$$

Total number of comparisons for the entire watershed computation (on an average case) is

$$LMD + LP \approx 3n \approx O(n) \quad (5)$$

3.3.3 Analysis of Conventional Hillclimbing Technique [14]

The number of comparisons for minima detection and flooding process (on an average) is $3n$. In addition, the lower-complete transformation involves n multiplications and n additions. Thus the computational complexity of the conventional watershed algorithm (on an average) is

$$= 3n\ (comparisons) + n\ (multiplications) + n\ (additions) \quad (6)$$

From the above analysis (Eqs. (5) and (6)), the proposed algorithm requires a total of $3n$ operations (only comparisons) as against an aggregate of $5n$ operations including $3n$ comparisons, $O(n)$ multiplications and $O(n)$ additions for the conventional algorithm [14].

4 Experimental Results

As discussed above, a considerable reduction in the computational complexity has been obtained. To avoid

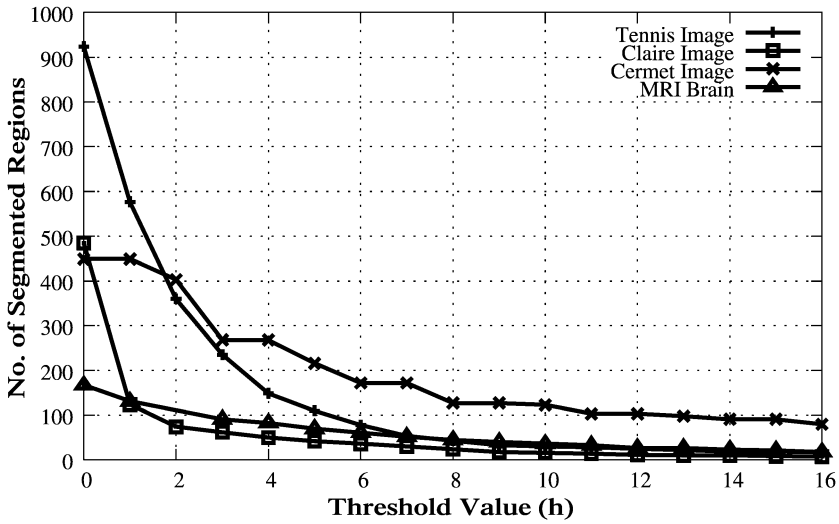

Figure 3 Number of regions for different threshold value (h).

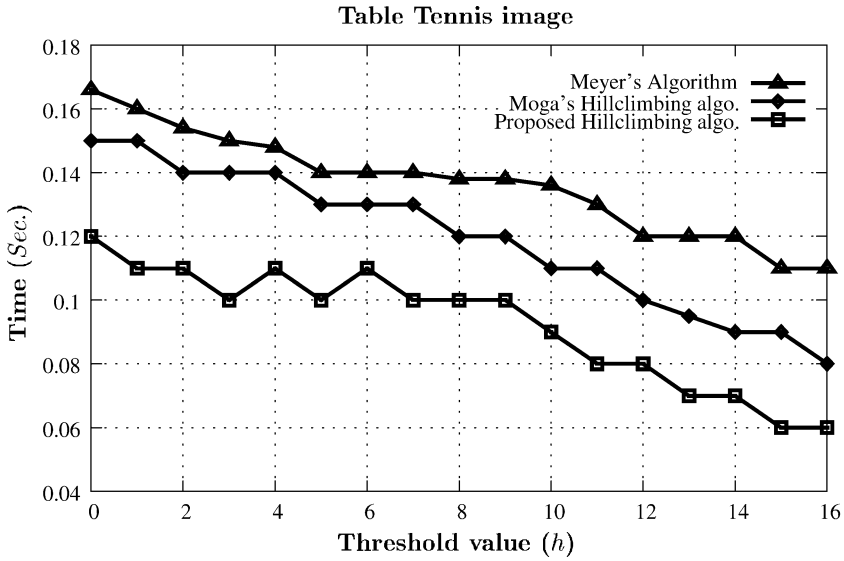

Figure 4 Simulation time while running Meyer's ordered queue based algorithm [12], Moga's Hillclimbing technique [14] and proposed hillclimbing technique on Tennis image.

over-segmentation, a morphological multi-scale gradient [27] image with different threshold values are used as the input for computing the watershed transform. The improved algorithm has been implemented on a Pentium IV 1.8 GHz computer under LINUX environment and applied on various test images to verify its effectiveness. Figure 2 shows the visual segmentation results of applying both Moga's algorithm [14] and the proposed algorithm on five standard images, namely Table Tennis, Cermet, Washington_ir, Steel Fissure and Blood Cells images. Comparison of the computation times of both algorithms is given in Table 1. The improved algorithm is faster and more efficient than the existing algorithm. Also, the segmentation results are obtained for different values of the threshold h required as a parameter in construction of the multi-scale gradient [27] image. Figure 3 shows that an increase in h leads to a reduction in the number of regions. Variation of simulation time with the values of h is plotted in Fig. 4.

Following a region-based performance evaluation scheme [28], the segmentation quality of the proposed algorithm is quantitatively assessed in terms of size and location of the segmented regions. Let S_1 and S_2 be the segmented image produced by proposed algorithm and its ground truth respectively, and expressed as

$$S_1 = \{R_1^1, R_1^2, R_1^3, \ldots, R_1^n\}$$
$$S_1 = \{R_2^1, R_2^2, R_2^3, \ldots, R_2^n\}$$

The directional Hamming distance from S_1 to S_2 is defined as

$$D_H(S_1 \Rightarrow S_2) = \sum_{R_2^i} \sum_{R_1^k \neq R_1^j, R_1^k \cap R_2^i \neq \emptyset} | R_2^i \cap R_1^k |,$$

where $|\ .\ |$ denotes the size of a set. Therefore, $D_H(S_1 \Rightarrow S_2)$ is the total area under the intersections between all $R_2^i \in S_2$ and their non-maximal intersected regions R_1^k's from S_1. The reverse distance $D_H(S_2 \Rightarrow S_1)$ can also be similarly computed. The region-based performance measure based on normalized Hamming distance is given as

$$p = 1 - \frac{D_H(S_1 \Rightarrow S_2) + D_H(S_2 \Rightarrow S_1)}{2 \times | S |},$$

Table 2 Quantitative evaluation of the segmented images.

Test Image	h	No. of regions	Mismatched pixels	Performance measure (P)
Claire (352×288)	10	17	23	0.99965
Tennis (352×288)	13	22	16	0.99984
Heart (256×256)	8	14	36	0.99945
Akiyo (164×144)	14	22	12	0.99949
MRI Brain (256×256)	12	38	31	0.99952
Cermet (256 × 256)	30	61	37	0.99943
Washinton (250×250)	16	46	61	0.99902
Steel Fissure (290×369)	15	10	18	0.99995
Blood Cells (212×353)	26	20	6	0.99992

Table 3 Comparison between simulation times and performance index of flooding-based method [26] and the proposed method.

Test image	Simulation time (s) for different methods		Performance index			
	Flooding method	Proposed method	Flooding method		Proposed method	
			No. of mismatched pixels	Performance measure	No. of mismatched pixels	Performance measure
Claire	0.11	0.075	539	0.99468	23	0.99965
Tennis	0.107	0.072	627	0.99266	16	0.99984
Heart	0.086	0.065	978	0.98648	36	0.99945
Akiyo	0.039	0.024	399	0.98278	12	0.99952
MRI Brain	0.039	0.024	1023	0.98181	31	0.99943
Cermet	0.082	0.064	714	0.98547	37	0.99943
Washinton	0.078	0.073	1205	0.97585	61	0.99902
Steel	0.102	0.091	264	0.99752	18	0.99995
Fissure BloodCells	0.074	0.065	572	0.99235	6	0.99992

where $|S|$ is the image size and $p \in [0, 1]$. The smaller the degree of mismatch, the closer the measure p is to one. The goal is to quantitatively describe the mismatch between S_1 and S_2. The performance measure P of the proposed algorithm evaluated for various test images is provided in Table 2, which shows that the homogeneous regions are identified by proposed algorithm and ground truth in an almost identical manner ($P \approx 1$).

4.1 Comparison with the Flooding-based Method

Table 3 lists the simulation time and performance index of flooding-based method [26] and the proposed method. As shown in Table 3, the execution time of the hillclimbing-based watershed algorithm proposed in the present paper is faster than that of the flooding-based method [26] because of the following reasons. The steepest lower distance

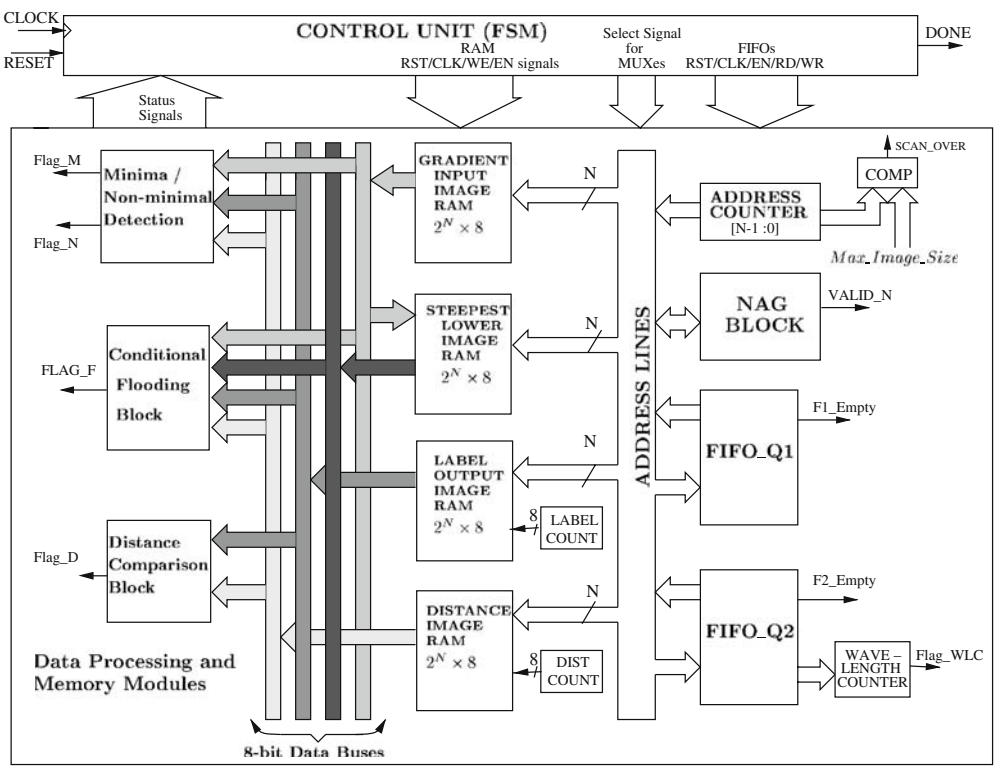

Figure 5 Complete architecture of the proposed watershed transform.

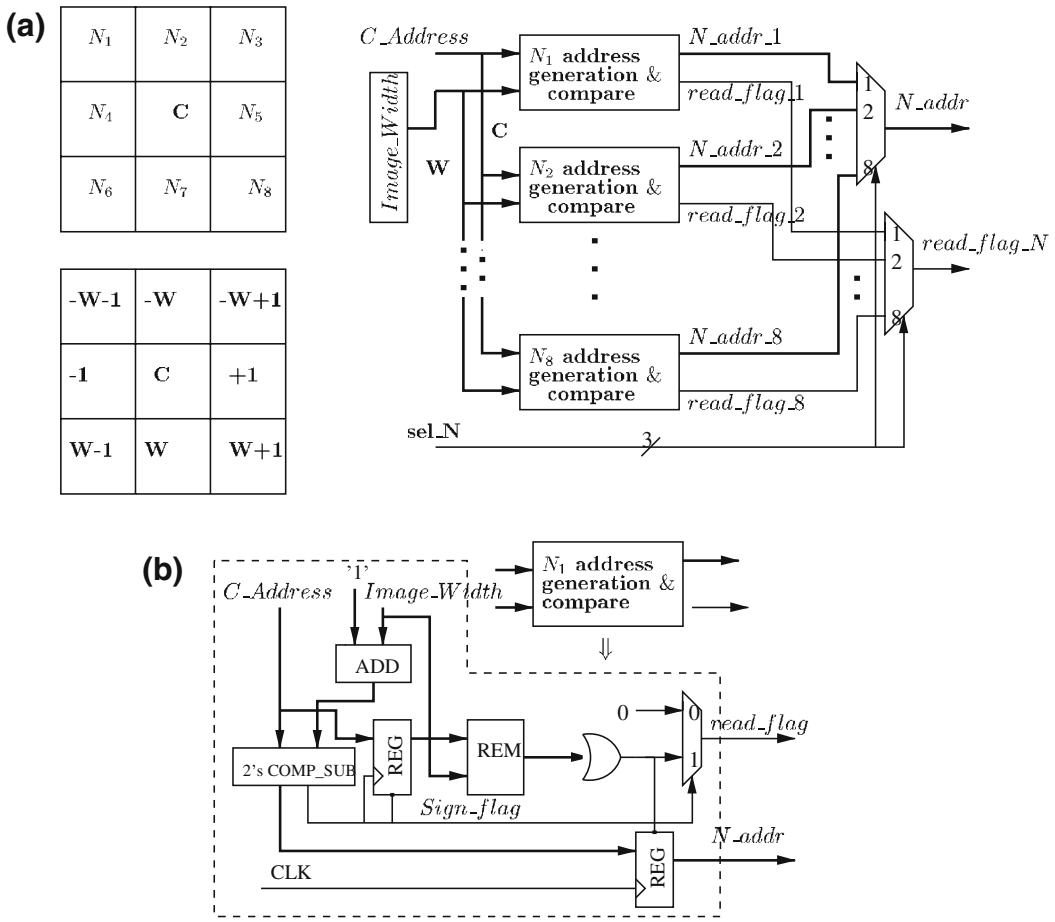

Figure 6 (a). The block diagram of the NAG block. (b). Schematic diagram for generating address of N_1.

transformation is introduced early in the proposed method, which leads to a decrease of the number of search points. Moreover, it facilitates synchronizing the process of propagating the labels in the non-minimal plateau, which has more than one closely located regional minima. On the other hand, the flooding-based technique [26] adopts a depth-first label propagation procedure, requiring sixteen additional supporting pixels for labeling the eight nearest neighbors of a central pixel under consideration. Thus, only one regional minimum is processed at a time. The drawback of the flooding-based method is that no label synchronization in label propagation is possible in a non-minima plateau (thick gradient) which has more than one closely located regional minimum. Out of the more than one possible local minima surrounding a non-minima plateau, one selects the one with the lowest value to flood the plateau in order to contain over-segmentation problem. Additionally, the proposed hillclimbing method requires three raster scans in contrast to four raster scans involved in the flooding-based method [26]. The degree of mismatch for the proposed method, as borne out by Table 3, is considerably less than that in case of the flooding-based method [26].

5 Proposed Prototype Architecture and its FPGA Implementation

This section presents a detailed discussion on a prototype architecture for implementing the improved hillclimbing technique proposed in this chapter. Moreover, an FPGA implementation of the architecture is briefly described.

5.1 Prototype Architecture

The block diagram of the architecture that implements the improved watershed algorithm is shown in Fig. 5. This architecture consists of four image RAM blocks, which are meant to hold the input image (GRADIENT INPUT IMAGE RAM), the output image of labels (LABEL OUTPUT IMAGE RAM), the steepest neighbor values (STEEPEST LOWER IMAGE RAM) and the lower distances (DISTANCE IMAGE RAM), two FIFO Queues, Minima/Non-minima Detection block, Distance Comparison block, Conditional Flooding block, four counters namely Address Counter (up), Label Counter (up), Distance Counter(up) and Wave_Length Counter(down), and a Control Unit. The control unit is a finite state machine

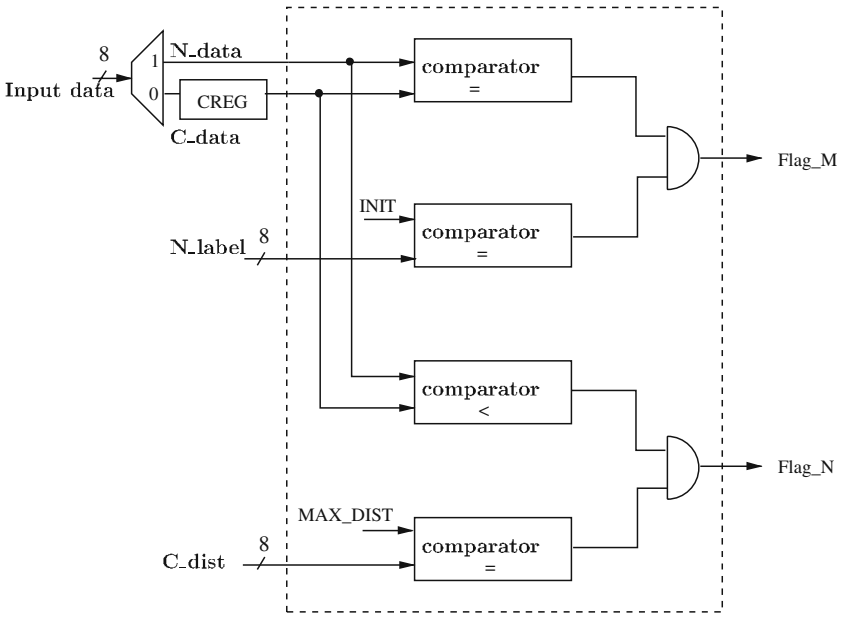

Figure 7 Minima/non-minima detection block.

(FSM) that controls the overall process including the initialization of RAMs and FIFOs, detection of Plateau of Minima and Non-minima, labeling of Local Minima and Computation of Lower Distance, and finally the Flooding process. Raster scan can be performed by using the Address Counter. A control signal scan_over is generated, when the Address Counter reaches the last pixel address (max_image_size) of image. The Wave_Length Counter is used to assign the proper distance during the analysis of the non-minima plateaus (or Lower Distance Computation process). Initially, it will be loaded with the contents of FIFO_Q2. The Wave_Length Counter is always decremented whenever a pixel is dequeued from FIFO_Q2. When the Wave_Length Counter reaches the value 0, a control signal flag_wlc is enabled. Description of the individual blocks follows next.

5.1.1 Neighbor Address Generation Block

The neighborhood addresses are generated in parallel through addition/ subtraction of the Image_Width W and the C_address as shown in Fig. 6. Binary addresses are used to generate the addresses of all the neighbors. The width of the image is known to the user and is stored in a register Image_Width. If the center pixel C in an image belongs to a boundary, some neighbor addresses may be incorrectly computed. Therefore, all these neighbor addresses are compared with the boundary conditions and the corresponding flags are generated, which shows the validity of the neighbors address. Through neighbor selection signal sel_N, any neighbor address can be selected.

The schematic diagram of the circuitry required to correctly generate the address of a pixel N_1 is given in Fig. 6b. The circuit blocks labeled ADD and 2's COMP_-SUB are used to generate the neighbor pixel address (N_Addr_1) from the center pixel address (C_Address). The block REM is a remainder circuit which is used to find the validity of the generated neighbor address. If the center pixel belongs to a boundary, the neighbor address may be incorrectly computed. To prevent this, the remainder left on dividing by the C_Address and the Image_Width is computed. The remainder determines the validity of the neighboring pixel (whether it is inside or outside the image) as the read_flag is generated.

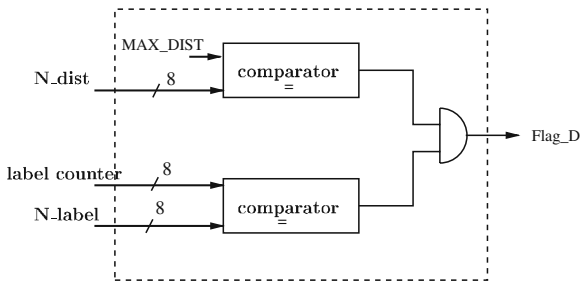

Figure 8 Distance comparison block.

5.1.2 Minima/Non-minima Detection

This block diagram as shown in Fig. 7 is used to detect whether the neighboring pixel belongs to a plateau or is a center pixel belonging to a non-minima plateau. Two control flags are generated on the basis of different conditions. First, the control unit stores the center pixel

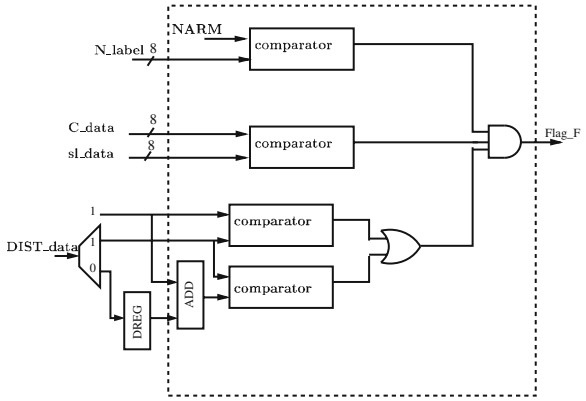

Figure 9 Conditional flooding block.

data of gradient image into a register CREG through the demux. In following clock cycles, the neighbor pixel data are available on the input_data and N_data inputs. The C_dist input always contains the modified center pixel distance.

The signal Flag_M is generated when the neighboring pixel data (N_data) of the gradient image and the label image (N_label) are equal to center pixel data (C_data) of the gradient image and the initialized value of label image INIT respectively. It indicates that the neighboring pixel belongs to its plateau and not visited previously. This status signal causes the control unit to generate the necessary signal to enqueue the address of the neighbor pixel into FIFO_Q1.

The signal Flag_N is generated when the neighbor pixel data (N_data) of gradient image is less than its center pixel data (C_data), and the center pixel data of distance image is equal to MAX_DIST. It indicates that the center pixel belongs to the non-minima plateau and it is an outer pixel. This status signal causes the control unit to generate the necessary signals to enqueue the address of the outer pixel into FIFO_Q2.

5.1.3 Distance Comparison Block

The distance comparison block shown in Fig. 8 detects whether the neighboring pixel belongs to a non-minima plateau for which no distance is assigned. The control signal Flag_D is generated when the neighboring pixel label data (N_label) and the distance image data (N_dist) are equal to the label counter value and the initialized value of the distance image MAX_DIST respectively. It indicates that the neighboring pixel belongs to a non-minima plateau and no distance is assigned to it. This signal makes the control unit generate the necessary signal to enqueue the address of the neighboring pixel into FIFO_Q2 for further exploration and assignment of the current distance value of the WAVE_LENGTH_COUNTER to the distance image pixel.

Table 4 Comparison between simulation time of hardware implementation and software implementation.

Test image (30 × 30)	Software time (msec)	Hardware time (μsec)
Claire	16	30.16
Tennis	16	29.04
Heart	12	28.74
Akiyo	10	28.45
MRI Brain	20	35.42
Cermet	20	34.96

Table 5 Design statistics for the entire architecture.

Hardware elements	No. of elements	% of utilization
Number of slices	489 out of 1,200	40
Total number slice registers	261 out of 2,400	25
Total number 4 input LUTs	827 out of 2,400	52
Number of bonded IOBs	18 out of 166	20
Number of block RAMs	5 out of 10	50

5.1.4 Conditional Flooding Block

The block diagram shown in Fig. 9 detects whether the neighboring pixel has the steepest arc to the center pixel or has lower distance equal to one more than that of the center pixel if both are on a plateau of non-minima. At first, the center pixel distance is assigned to a register DREG through Demux. In the next clock cycle, the neighbor pixel data is compared with the center pixel data as N_data with NARM, sl_data with C_data, and dist_data must be 1 or equal to DREG +1. A control signal flag (Flag_F) is generated when the above condition is met, which decides the propagation of the label of the center pixel to its neighbors.

5.1.5 Control Unit (FSM)

The control unit (FSM) generates the reset signal to all the blocks as shown in the complete architecture of Fig. 5. Moreover, it generates the signals to control the MUXex

Table 6 Design statistics for the entire architecture.

Timing summary	
Minimum period	20.056 ns
Maximum frequency	49.86 MHz
Maximum combinational path delay	17.152 ns
Maximum net delay:	8.65 ns

and the DEMUXes, read, write, enable signals for FIFOs, RAMs and load, enable, reset signal for counters on the basis of the control signals generated through "Detection of Minima/Non-minima" block, "Distance Comparison" block, "Conditional Flooding" block and the signals generated through FIFOs (f1_empty, f2_empty). After computing the watershed segmentation, a DONE signal is set to 1.

5.2 FPGA Implementation

The proposed architecture described in section 5.1 has been described in VHDL [29, 30] and simulated by Modelsim, and synthesized under the Xilinx ISE4.1i system targeted to virtex FPGA series device. The proposed architecture has been simulated for various test images of size (30×30). An image of size greater than 30 × 30 requires more than one chip for its implementation. Due to limitations of the memory on the virtex device, the FPGA implementation has been limited to an image of size 30 × 30. In this design, the block RAMs are configured with 8-bit data lines and 9-bit address lines. For these address lines a 9-bit up counter is required to generate addresses or ADDRESS COUNTER, and other counter like LABEL COUNTER, DISTANCE COUNTER are configured as 8-bit up counters.

The entire design has been simulated on ModelSim Xilinx 5.5b. In the first clock cycle, a reset signal has been applied on an input pin of FPGA. In the second cycle, the control unit (FSM) will reset all counters, block RAMs and FIFOs. Also a DONE signal is set to 0 by the control unit. From the third cycle onwards, data on DIN pin will be stored in GRADIENT INPUT IMAGE DATA RAM corresponding to the address generated by the Address Counter. A Scan_Over control signal will become 1 when Address Counter reaches the last address (MAX_IMAGE_SIZE) of the image. The control signal DONE is set to 1 when the entire watershed computation is completed.

Tests were run assuming a clock frequency of 50 MHz. The simulation results obtained for various test images are given in Table 4. From the simulation results, the hardware implementation is seen to be faster than the software version by an order of 3. The software results shown in Table 4 are achieved using Linux environment on a Pentium IV 1.8 GHz processor system. The entire design has been simulated, synthesized at gate level and then implemented on the device XCV100-6PQ240. The performance of the FPGA implementation is limited to 49.86 MHz in XCV100-6PQ240. By using sharing of resources such as FIFOs, counters, muxes and demuxes, and by employing a single FSM for the control unit, this design has less hardware requirement compared to the architecture [26] required to implement the improved flooding-based watershed algorithm. FPGA design summary of the entire architecture discussed in Section 5 is accommodated in Tables 5 and 6. The design utilizes almost 40% of CLBs and 25% of Registers.

6 Conclusions

The present paper proposes an improved watershed transform method based on hillclimbing simulation and its prototype architecture. By avoiding time-consuming lower complete transformation, the proposed algorithm has been shown to perform better for same input images compared to the original algorithm due to Moga [14]. Computation of the lower complete image is avoided in the present algorithm at the cost of two additional comparisons in the flooding process. Further reduction of the overall computation time has been achieved by adopting some other modifications, namely employment of two FIFO queues and computation of the steepest lower neighbor image during the process of detection of minima/non-minima plateau. The paper analyzes the complexity of then proposed algorithm, which requires only $3n$ comparison operations as against an aggregate of $5n$ operations including $3n$ comparisons, n multiplications and n additions required by the conventional algorithm [14], where n is the dimension of the image. A quantitative measure of accuracy of the segmentation results produced on various images by the proposed algorithm has been provided. The paper also describes a prototype hardware architecture for an effective realization of the proposed algorithm. The architecture has been synthesized in an appropriate FPGA environment. Relevant design statistics of the FPGA implementation of the architecture are given.

Acknowledgement The authors are grateful to the anonymous reviewers for their constructive suggestion on improvement of the paper.

References

1. Haralick, R. & Shapiro, L. (1985). Image segmentation techniques. *Computer Vision, Graphics, Image Processing, 29*(1), 100–132, January.
2. Pal, N. R., & Pal, S. K. (1993). A review on image segmentation techniques. *Pattern Recognition, 26*(9), 1277–1294.
3. Pratt, W. K. (2003). *Image segmentation* (3rd Ed., ch. 17 pp. 551–588). John Willey and sons, INC.
4. Weska, J. S. (1978). A survey of threshold selection techniques. *Computer Graphics and Image Processing, 7*(2), 259–265, April.
5. Mardia, K. & Hainsworth, T. (1988). A spatial thresholding method for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 10*, 919–927.
6. Sankur, B., Abak, A. T., & Baris, U. (1999). Assessment of thresholding algorithms for document processing," in *Proceedings of IEEE International Conference on Image Processing, 1* (pp. 580–584). Japan: Kobe, October.

7. Prager, J. M. (1980). Extracting and labeling boundary segments in natural scenes. *IEEE Transactions on Pattern Recognition and Machine Intelligence, 2*(1), 16–27.
8. Perkins, W. A. (1980). Area segmentation of images using edge points. *IEEE Transactions on Pattern Recognition and Machine Intelligence, 2*(1), 8–15.
9. Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8*, 679–698.
10. Beaulieu, J. M., & Goldberg, M. (1989). Hierarchy in picture segmentation: a stepwise optimization approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 11*(2), 150–163.
11. Wu, X. (1993). Adaptive split-and-merge segmentation based on piecewise least-square approximation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 15*, 808–815, August.
12. Beucher, S. & Meyer, F. (1993). The morphological approach to segmentation: The watershed transformation. In E. R. Dougherty (Ed.), *Mathematical morphology in image processing* (pp. 433–481). New York: Marcel Dekker Inc.
13. Chang, Y. L., & Li, X. (1994). Adaptive image region growing. *IEEE Transactions on Image Processing, 3*(6), 868–873.
14. Moga, A. N. (1997). Parallel watershed algorithms for image segmentation. Ph.D. dissertation, Tampere University of Technology, Tampere, Finland, February.
15. Roerdink, J. B. T. M., & Meijster, A. (2001). The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta Informaticae, 41*, 187–228.
16. Derin, H., & Elliott, H. (1987). Modeling and segmentation of noisy and textured images using gibbs random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 9*, 39–55.
17. Srinivas, C., & Srinath, M. D. (1989). Compound gaussian markov random field model for image segmentation," in *International Conference on Acoustics, Speech, and Signal Processing, vol. 3* (pp. 1586–1589). UK: Glasgow, May.
18. Dubes, R. C., Jain, A. K., Nadabar, S. G., & Chen, C. C. (1990). MRF model-based algorithms for image segmentation," in *Proceedings of 10th International Conference on Pattern Recognition, 1* (pp. 808–814). Atlantic City, NJ, USA, June.
19. Kuria, T. (1991). An efficient agglomerative clustering algorithm using a heap. *Pattern Recognition, 24*(3), 205–209.
20. Ohm, J. R., & Ma, P. (1997). Feature-based cluster segmentation of image sequences. in *Proc. IEEE international Conference on Image Processing* (pp. 178–181).
21. Pauwels, J., & Frederix, G. (1999). Finding salient regions in images. *Computer Vision and Image Understanding, 75*, 73–85.
22. Vincent, L., & Soillé, P. (1991). Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 13*(6), 583–598.
23. Meyer, F., & Beucher, S. (1990). Morphological segmentation. *Journal of Visual Communication and Image Representation, 1* (1), 21–45.
24. Meyer, F. (1994). Topographic distance and watershed lines. *Signal Processing, 38*(1), 113–125, July.
25. Viero, T. (1996). Algorithms for image sequence filtering, coding and image segmentation. *Ph.D. dissertation, Tampere University of Technology.* Finland: Tampere, January.
26. Rambabu, C., Chakrabarti, I. & Mahanta, A. (2004). A novel flooding-based watershed algorithm and its prototype hardware architecture. *IEE Proceedings - Vision, Image and Signal Processing, 151*(3), 224–234, June.
27. Wang, D. (1997). A multiscale gradient algorithm for image segmentation using watersheds. *Pattern Recognition, 30*(12), 2043–2052, January.
28. Huang, Q., & Dom, B. (1995). Quantitative methods for evaluating image segmentation. in *IEEE International conference on Image Processing, 3* (pp. 53–56). Washington D. C., October.
29. Bhaskar, J. (1999). *VHDL primer* (3rd Ed). Prentice Hall.
30. Perry, D. L. (2001). *VHDL* (3rd Ed). Tata McGraw-Hill Publishing Company Limited.

C. Rambabu received his B.E degree in Electronics and Communication Engineering from AU, India in 1995, M. Tech in Automation and Computer Vision from E&ECE, IIT Kharagpur, India in 1998 and Ph.D. from IIT Guwahati, India in 2005. From 2005 to 2006, he worked as a Research Fellow in UVR Lab, GIST, South Korea. He is currently employed as a Research Fellow at Bioinformatics Institute (BII), Imaging Group, Singapore. His areas of interest are computer vision, image/video processing, Multi-dimensional Microscopic image analysis and VLSI Signal processing. He has published several papers in international journals and conferences in these areas.

Indrajit Chakrabarti received his B.E. and M.E. degrees in Electronics and Telecommunication Engineering from Jadavpur University, India in 1987 and 1990. Subsequently, he received Ph.D. from Indian Institute of Technology (IIT) Kharagpur, India in 1997. From 1998 to 2004, he worked as an Assistant Professor and later as an Associate Professor in the Department of Electronics and Communication Engineering, IIT Guwahati. He is presently serving as an Associate Professor in the Department of Electronics and Electrical Communication Engineering, IIT Kharagpur. His areas of interest include VLSI architectures for image processing, digital signal processing and telecommunication.