AN IMPROVED FLOODING BASED WATERSHED ALGORITHM

Ch. Rambabu

ECE Department, IIT Guwahati.
rambabu@iitg.ernet.in

I. Chakrabarti and A. Mahanta

ECE Department, IIT Guwahati.
indra@iitg.ernet.in, anilm@iitg.ernet.in

ABSTRACT

This paper presents a fast watershed algorithm derived from Meyer's simulated flooding based watershed algorithm by ordered queues. The proposed algorithm uses a single queue and conditional neighborhood comparisons for processing 3×3 neighboring pixels to reduce computational complexity while compared to the conventional watershed algorithm. Computational complexity of the principal steps of the proposed algorithm is analyzed. Simulation results of running the proposed algorithm and the conventional algorithm on different images are given for comparison.

1. INTRODUCTION

Image segmentation involves partitioning of a given image into a number of homogeneous segments. It may also be considered as a pixel labeling process in the sense that all pixels that belong to the same homogeneous region are assigned the same label. Among the existing algorithms [1, 2, 3], watershed transformation has proved to be a very useful and powerful tool for morphological image segmentation because of its moderate computational complexity and its ability to identify the important closed contours of a given image. Watershed transformation has been used in several applications like object-based motion estimation (MPEG-4), biomedical and medical image processing and computer vision amongst others.

In watershed transformation, a gradient image obtained by applying an appropriate morphological gradient operator on a gray scale image is considered as a topographic surface where the gray scale values of a pixel denotes the altitude of that pixel. Each pixel in this digital image is assigned a label during its assignment to the catchment basin of a regional minimum. One group of watershed algorithms aim to detect the catchment basins by simulated flooding process[4, 5]. Among these, there are algorithms which construct the watershed lines [4] and those which do not delimit the basins by watersheds (0-width watershed lines) [5]. The algorithm introduced by Vincent and Soille [4] examines all the pixels in an image at successive gray scale values. Meyer's algorithm [5] is based on a definition of flooding by the watersheds and a two-level ordering relation provided by an ordered queue(OQ). Several shortest-path algorithms for the watershed transformation based on topographical distance can also be found in the literature [3].

In this paper, we proposed an improved watershed algorithm that is analogous to Meyer's simulated immersion algorithm based on ordered queue. In Meyer's algorithm [5] simulation of the flooding process involves 256 queues. The proposed scheme uses only one queue and decides the labels of neighboring pixels based on conditional comparisons. It demonstrates reduced computational complexity and almost identical performance when compared to Meyer's algorithm.

The organization of this paper is as follows. Sections 2 and 3 describe the proposed watershed algorithm and simulation results respectively. Section 4 concludes this paper.

2. THE PROPOSED WATERSHED ALGORITHM

Meyer's algorithm consists of two main stages: detection and labeling of the regional minima, and flooding or simulated immersion. The algorithm starts by detecting and labeling of the initial seeds that is, the minima of the gradient, which characterize the regions of interest. The latter are defined as connected plateaus

of pixels in gradient image which do not have neighboring pixels of lower gray level (plateau of minima). Starting from the minima, the recursive label propagation (flooding) is performed using an ordered queue (H FIFO queues), where ($0 \leq H \leq 255$) one queue is meant for each of the H gray levels in image.

The label propagation(flooding) of Meyer's algorithm is illustrated in Figure 1 in which C is the center pixel and N_1 - N_8 are its 8-connected nearest neighbors. The ordered queues (H FIFOs) are initialized with the labeled regional minimas which have at least one non-labeled neighboring pixel. Pixels are dequeued from the current FIFO queue, one at a time; next, their labels are assigned to the non-labeled neighboring pixels of higher altitude, which are in turn inserted in the corresponding FIFO queues. When all the queues have been emptied, it signifies that each pixel in the image has been labeled and the flooding procedure stops. The label propagation require 256 queues, which makes the hardware realization of Meyer's algorithm complex and costly.

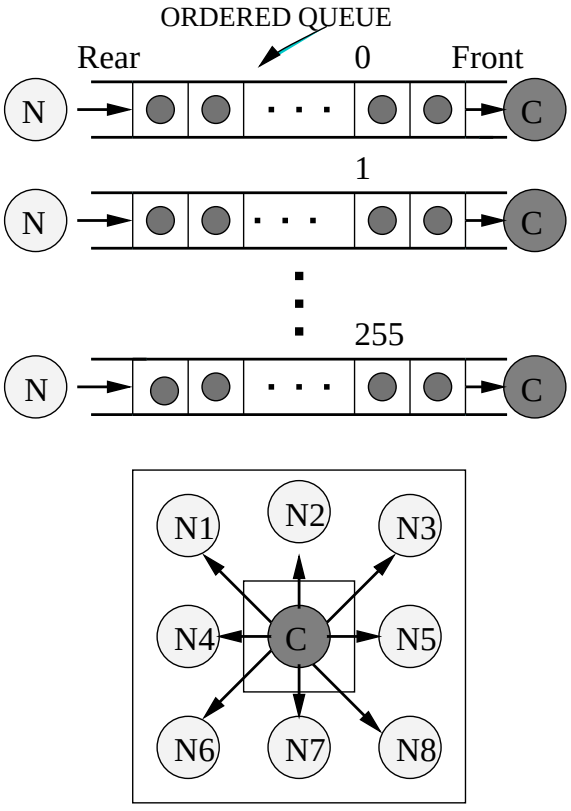

Figure 1: *Label propagation in Meyer's algorithm*

In our proposed scheme, the identification of local minima and their labeling remains the same. The label propagation which decides the labels of neighboring (N_1–N_8) pixels of center pixel C but requires 16 additional supporting pixels(S_1–S_{16}), is illustrated in Figure 2. Only one single queue however, is used as against 256 queues in Meyer's label propagation. In the first step, one detects the labeled regional minimas which have at least one non-labeled neighboring pixel and stored in an array LM_ARRAY. Next, one initializes the queue LP_FIFO with the array of labeled regional minimas one at a time. Pixels are dequeued from the queue LP_FIFO and one attempts to decide the labels of neighboring pixels (N_1-N_8) based on the conditional neighborhood comparisons. Those pixels, whose labels can be decided, are inserted into the FIFO queue. The recursive label propagation from the present local minima stops when the FIFO queue is empty. Next, another local minima from the array is put into the queue LP_FIFO. When all the local minimas of LM_ARRAY have been visited, it signifies that each pixel in the image has been labeled and the entire flooding procedure stops.

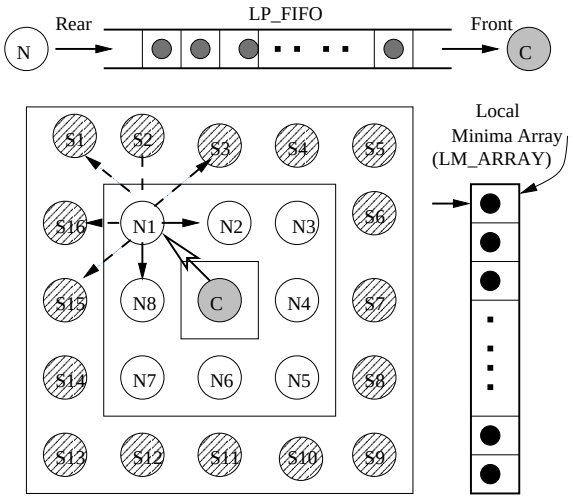

Figure 2: *Label propagation in proposed algorithm*

To decide the label of a pixel N_i, i = 1,...,8, the 4-connected neighbors of N_i (which are also neighbors of center pixel C) and the 8-connected supporting pixels of N_i are considered and processed in different conditional steps. For example, to find the label for N_1, at the first step we need to consider N_1 and two of its 4-connected neighbors namely N_2 and N_8 which would

be compared with the center pixel C. If the condition (viz. the gray level of C being lower than or equal to that of N_2, N_8) is satisfied, then one would consider the 8-connected supporting pixels namely S_1, S_2, S_3, S_{16} and S_{15} which would be compared with C as shown in Fig. 2. The details of the two major phases of the proposed algorithm are follows.

Local Minima Detection and Labeling: At this stage, the minimum plateaus are first detected and labeled by a raster scan of the input gradient image G. For each not yet visited pixel p, its 4-connected or 8-connected neighborhood is inspected. Thus if all the neighbors are of higher gray level than p, then p is deemed to be a local minimum, and a label is assigned to it; otherwise, it is labeled as NARM(not a regional minima). If p is found to have a neighbor with the same gray level as itself, it means p is on a plateau, which is further explored and a breadth-first scan is carried out to propagate the label of p throughout the plateau. The pseudo-code of the minimum detection and labeling procedure is as follows.

```
Input: Morphological gradient image G=(D)
Output: Labeled Image L=(D)
INIT ← -1; NARM ← -2; MASK ← -3
fi fo_init(Q);curlab ← 0
for all p ε D do /* D: domain of the image */
  L[p] ← INIT
end for
for all p ε D with L[p]=INIT do
  L[p] ← MASK; fi fo_add(p,Q); fl ag ← 0
  /* plateau detection phase */
  while not fi fo_empty(Q) do
    q ← fi fo_remove(Q)
    for all r ε N_G^+(q) do
    /* N_G^+-forward raster scan neighboring pixels*/
      if G[r] < G[q] then
        fl ag ← 1
      elsif (G[r] = G[q]) and (L[r]=INIT) then
        L[r] = MASK; fi fo_add(r,Q)
      end if
    end for
  end while
  /*plateau analysis phase */
  if fl ag = 1 then
    L[q] = NARM; fi fo_add(q,Q)
  else
    curlab ← curlab + 1; L[q] = curlab;
    fi fo_add(q,Q)
  end if
  /* plateau labeling phase */
  while not fi fo_empty(Q) do
    q = fi fo_remove(Q)
    for all r ε N_G^-(q) do
    /*N_G^- backward raster scan neighboring pixels*/
      if L[r] = MASK then
        L[r] = L[q]; fi fo_add(r,Q)
      end if
    end for
  end while
end for/* end of the local minima detection */
```

Simulated Flooding: At this stage, one first detects the labeled local minimas and places them in an array S. Initialize the queue FQ with the local minima (from queue S) which has at least one non-labeled neighborhood pixel. Pixel p is dequeued from the queue FQ one at a time and its label is assigned to the non-labeled neighboring pixel q by using conditional comparisons among the neighbors of higher altitude. Next q is inserted into queue FQ. Flooding process stops when all the elements in the array S have been visited. The pseudo-code of the flooding process is as follows.

```
Input: Morphological gradient image G=(D)
Output: Labeled Image L=(D)
fi fo_init(FQ)
Store the labeled local minimas in an array S=(M)
for all m ε M do
  fi fo_add(S[m],FQ)
  while not fi fo_empty(FQ) do
  p ← fi fo_remove(Q)
  for all q ε N_G(p) do
    fl ag_1 ← 1; fl ag_2← 1
    if L[q] < 0 then
  /* Compare N_G^4 the 4-connected neighbors of p */
    for all r ε (N_G^4(q) ∧ N_G(p)) do
      if G[r] < G[p] then
        fl ag_1 ← 0
      end if
    end for
    if fl ag_1 = 1 then
    /* Compare 8-connected neighbors of q
    which are not neighbors of p */
    for all r ε (N_G(q) -((N_G(q) ∧ N_G(p))
              ∨ {p})) do
      if G[r] < G[p] then
        fl ag_2 ← 0
      end if
    end for
    if fl ag_2 = 1 then
        L[q] ← L[p]; fi fo_add(q,FQ)
    end if
    end if
    end if
  end for
  end while
end for /* end of the simulated fl oding phase*/
```

Complexity Analysis: Let M×N be the dimension of the image G having P plateaus.
Local minima detection and labeling: let p_1, p_2, p_3, ... p_P be P plateaus, each p_i having n_i pixels.
Total number of comparisons for local minima detection and labeling is
$= (\sum_{i=1}^{P} n_i) * N_G^+ + (\sum_{i=1}^{P} n_i) * N_G^8$
$+ (\sum_{i=1}^{P} n_i) * N_G^-$
Simulated flooding: let l_1, l_2, l_3, ... l_L be L local minima plateaus in the image G; moreover, let l_j have m_j pixels which have at least one NARM (Not a regional minima) neighborhood pixel. Let C be the present center pixel and N_1, N_2 ... N_8 the 8-connected neighbors of the center pixel.
The number of comparisons for deciding the labels of neighboring pixels (in the worst case) is
$= N_G^8(C) + \sum_{i=1}^{8}(N_G^4(N_i) \wedge N_G^8(C))$
$+ \sum_{i=1}^{8}(N_G^8(N_i) - ((N_G^8(N_i) \wedge N_G^8(C)) \vee \{C\}))$
Total number of comparisons for entire label propagation (in the worst case) is then
$= (MN)/2 * \sum_{j=1}^{L} m_j * (N_G^8(C)$
$+ \sum_{i=1}^{8}(N_G^4(N_i) \wedge N_G^8(C))$
$+ \sum_{i=1}^{8}(N_G^8(N_i) - ((N_G^8(N_i) \wedge N_G^8(C)) \vee \{C\})))$

3. SIMULATION RESULTS

The proposed algorithm has been implemented under Linux 7.2 environment on a Pentium IV computer and applied on various test images to verify its effectiveness. Table 1 shows that the proposed algorithm is faster than Meyer's algorithm. Also, the segmentation results are obtained for different values of threshold h involved in multi-scale gradient operation[6]. Figure 3 shows that an increase in h leads to reduction in the number of regions. Variation of simulation time with the values of h is plotted in Figure 4. Results of running Meyer's algorithm and the proposed algorithm on a multi-scale morphological gradient[6] version of 352×288 Tennis image (as shown in Figure 6) appear in Figure 7 and Figure 8 respectively, which show that the homogeneous regions are identified by Meyer's algorithm and the proposed watershed algorithm equally well.

Test Image	Thr. value	No. of regions	Meyer's algo.	Prop. algo.
Claire 352×288	10	17	125	110
Tennis 352×288	13	22	120	107
Heart 256×256	8	14	93	86
Akiyo 164×144	14	22	36	39
MRI Brain 256×256	12	38	92	84

Table 1: *Simulation time (in milliseconds) results for Meyer's and the proposed algorithm.*

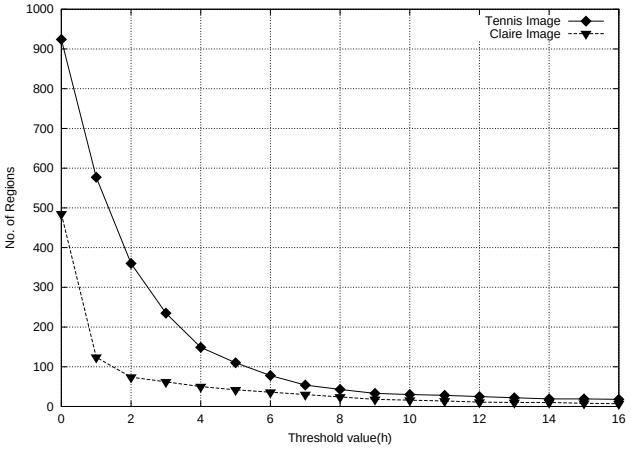

Figure 3: *Number of regions for different threshold value(h).*

4. CONCLUSIONS

The present-day image analysis algorithms incorporate watershed algorithm as an important pre-processing step. The present paper reports a new improved technique of simulated flooding based watershed algorithm. The proposed algorithm exhibits similar performance at reduced computational and hardware complexity while compared to Meyer's algorithm.

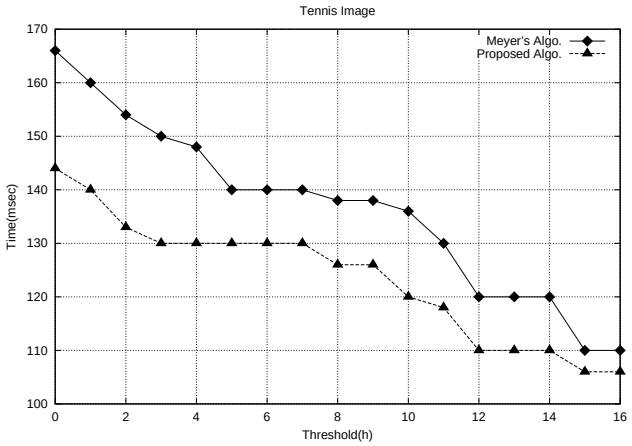

Figure 4: *Simulation time while running Meyer's and proposed algorithms on Tennis image*

REFERENCES

[1] R. Haralick and L. Shapiro, "Image segmentation techniques," *Computer Vison, Graphics, Image Processing*, vol. 29, pp. 100–132, 1985.

[2] N. Pal and S. Pal, "A review on image segmentation techniques," *Pattern Recognition*, vol. 26, pp. 1277–1294, 1993.

[3] Jos B. T. M. Roerdink and Arnold Meijster, "The Watershed Transform: Definitions, Algorithms and Parallelization Strategies," in *Fundamenta Informaticae*, vol. 41, pp. 187–228, IOS Press, 2001.

[4] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE PAMI*, vol. 13, no. 6, pp. 583–598, 1991.

[5] S. Beucher and F. Meyer, "The morphological approach to segmentation: the watershed transformation," in *Mathematical Morphology in Image Processing*, pp. 433–481, New York : Marcel Dekker Inc., 1993.

[6] D. Wang, "A multiscale gradient algorithm for image segmentation using watersheds," *Pattern Recognition*, vol. 678, no. 12, pp. 2043–2052, 1997.

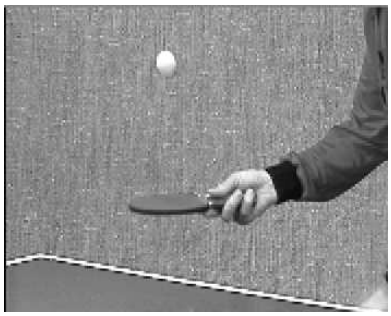

Figure 5: *Original Tennis image.*

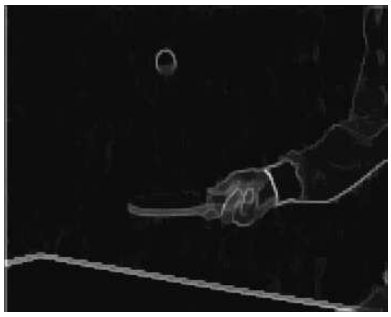

Figure 6: *Morphological multi-scale gradient operation on Tennis image with a threshold value of 13.*

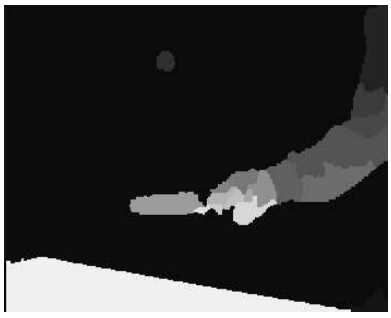

Figure 7: *Segmentation produced by Meyer's watershed algorithm on Tennis image.*

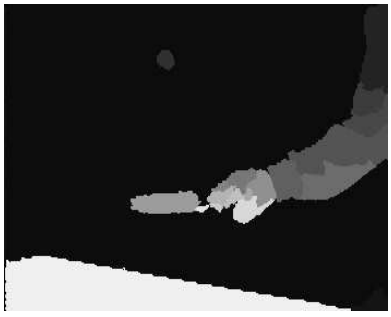

Figure 8: *Segmentation produced by the Proposed watershed algorithm on Tennis image.*