

Parallel Watershed Algorithm Based on Sequential Scanning*

Alina N. Moga, Timo Viero, and Moncef Gabbouj
Signal Processing Laboratory
Tampere University of Technology
P.O. Box 553, FIN-33101 Tampere, Finland

Michael Nölle, Gerald Schreiber, and Hans Burkhardt
Technical University of Hamburg-Harburg
Harburger Schloßstraße 20, D-21071 Hamburg, Germany

ABSTRACT — The watershed transformation is widely used in morphological image segmentation [1], e.g., in industrial, and biomedical applications, where digital pictures of 512 x 512 pixels, 1K x 1K, or even larger are not uncommon. Therefore, large amount of data and, consequently, complex analysis, entail parallel algorithms. In this paper, a fast SPMD (Single Program Multiple Data) watershed algorithm based on sequential scanings is rendered. The task performed by the present algorithm is to compute the watershed image by integrating the morphological gradient of the original image. The technique, iterative by nature, is implemented on a multitransputer system (with the Bruijn interconnecting network) by repeated scans and message passing among processors until the computation stabilizes. The algorithm is well suited for SIMD (Single Instruction Multiple Data) computers since no ordered queues (see [1, 4]) are used. Speedup evaluates the quality of the parallel algorithm.

1 Introduction

Watershed transformation is a segmentation algorithm applied on gray-scale images. A more compact image representation is provided by splitting the morphological gradient of an image into geodesic influence zones.

The parallelization strategy is based on the distribution of the image in a chessboard fashion into $m \times n$ subimages, corresponding to a mapping of the application to $m \times n$ processes. In order to perform computation in all pixel locations within the distribution subimages, each subimage is extended with an area of one grid-point width, such that the underlying subgrids are overlapping (one row/column/point from the adjacent subimage, if any, otherwise, fictious pixels with very high intensity-255).

Our watershed algorithm is based on the flooding scheme used by F. Meyer [3] in which flooding

is done by image integration [7]. The algorithm performs repeated raster and antiraster scans in the image, i.e. it is a sequential algorithm [6]. By assigning to the pixels of the regional minima values equivalent to their values in the original image as limit conditions, integration of the gradient image reproduces the original image [2, 3]. Image integration is essentially used for searching of minimal paths between regional minima and all other pixels, where the length of a path is given by the topographic distance [3]. Each pixel belongs to a shortest path originating from a minimum, from where a label is propagated along the whole path. Pixels on paths originating from the same minimum will get the same label, representing a distinct zone in the output image.

The remainder of the paper is organized as follows. In Section 2 commonly used definitions are introduced. Section 3 presents the parallel watershed algorithm in detail. Simulations and speedup evaluations are presented in Section 4.

2 Preliminary Definitions

- $D_f \subset \mathbf{Z}^2$ denotes the *domain* of a two-dimensional (2-D) digital gray-scale image f ;
- $N_G(p)$ denotes the *neighboring pixels* of a pixel p with respect to a given grid $G \subset \mathbf{Z}^2 \times \mathbf{Z}^2$;
- If we consider a global domain D_f split to equal size, disjoint subdomains $D_{f_i^a}$ -*distribution subdomains*- i.e. $D_f = D_{f_1^a} \cup D_{f_2^a} \cup \dots \cup D_{f_{m \times n}^a}$ and $D_{f_i^a} \cap D_{f_j^a} = \emptyset, i \neq j$, then $D_{f_i} = \{p | p \in D_f, N_G(p) \cap D_{f_i^a} \neq \emptyset\}$ are the *overlapping subdomains*, and the associated subimages f_i are overlapping too. $D_{f_i} \setminus D_{f_i^a}$ denotes the extension area;
- The *neighboring subimages* $N_f(f_i)$ of a subimage f_i are those subimages f_j such that $D_{f_i} \cap D_{f_j} \neq \emptyset$. In addition, if f_i belongs to the workspace of the process P_i , then $N_G(P_i) = \{P_j | f_j \in N_f(f_i)\}$ denotes the set of the *neighboring processes* of P_i ;
- A *boundary* between two neighboring subimages f_i and f_j is $B(f_i, f_j) = D_{f_i} \cap D_{f_j}$, and the *edge*

*This research has been supported in part by the European Communities Program Esprit III BRA NAT No. 7130

of the distribution subimage of f_i with the same neighboring subimage f_j is $E(f_i, f_j) = D_{f_i} \cap D_{f_j}$. $B(f_i, f_j)$ acts as a replica in P_i 's workspace of the edge $E(f_j, f_i)$ of the subimage f_j stored in the adjacent process P_j ;

- $\partial(p)$ equal to 0 if $\forall p' \in N_G(p) f(p') > f(p)$, and $\max\{f(p) - f(p') \mid p' \in N_G(p), f(p') \leq f(p)\}$, otherwise, defines the *morphological gradient* of an image f [3, 7];
- The *lower distance* function d is defined as follows: $d(p) = 0$ if p is a minimum; else, $d(p)$ is the length of the shortest path $P = \{p = p_0, p_1, \dots, p_s = q\}$ between pixels p and q such that $f(q) < f(p)$, $f(p_i) = f(p)$, $0 \leq i \leq s - 1$;
- The *lower-complete* function l is defined by $l(p) = \lambda \times f(p) + d(p) \forall p \in D_f$, where $\lambda = \max\{d(p) \mid p \in D_d\} + 1$ denotes the *largest lower distance* in d .

3 Description of the Parallel Watershed Algorithm Based on Sequential Scanning

The parallel watershed algorithm here introduced performs in four stages: detection of the minima pixels, computation of the lower-complete image, labeling the minima, and flooding by image integration. Each of these stages are next presented.

3.1 Detection of the minima pixels

The key of the watershed computation is the integration of the gradient image. The sources of the integration are the regional minima, and their values act as integration constraints. Therefore, minima are first detected inside each subimage by computing the lower distance subimage, initialized with MAXDIST ($= \infty$) in every pixel location. Then, in a raster and antiraster scan in each distribution subdomain, the following corrections are made for each pixel p according to all pixels p' in the past neighborhood (already scanned) of p denoted by $N_G^-(p)$:

- (1) if $f_i(p') < f_i(p)$ and $d_i(p) > 1$
 $d_i(p) \leftarrow 1$;
- (2) else if $f_i(p') = f_i(p)$ and $d_i(p) > d_i(p') + 1$
 $d_i(p) \leftarrow d_i(p') + 1$.

Unfortunately, a single raster and antiraster scan is not enough to set the correct lower distance for some plateaus of non-minima (spiral shaped constant region embedded in higher altitude areas). Consequently, raster and antiraster scanings are repeated, until no changes occur. The task performed in this loop is to adjust the distances insides connected plateaus. Therefore, step (1) in the above described loop is not comprised in these subsequent loops.

However, because the global domain is distributed, it is possible for plateaus of non-minima

to extend over several subdomains. In order to compute the global shortest paths, every process P_i sends to each process $P_j \in N_G(P_i)$ the distance values of pixels within the edge $E(d_i, d_j)$. At its turn, P_i stores similar information coming from each neighboring process P_j on the boundary $B(d_i, d_j)$. Then, a new loop consisting of raster and antiraster scans is performed as above to update the distance values inside the distribution subdomain D_{d_i} , according to the data in the extension area. Again, a single communication iteration may not be enough. Consider the case of a plateau of non-minima spread over several subdomains, and which has a lower neighbor exactly in one subdomain. After one communication step, the correct lower distance will be available only in the subdomain where the lower pixel lies and the adjacent subdomains which contain parts of the plateau. Changes in the edges of the distribution subdomains must be communicated again to the neighboring subdomains. The above iteration, consisting of communication and update, is repeated until the computation stabilizes, such that changes are propagated within the entire image. The moment of termination is detected by a master process which supervises all the processes.

At the end of this stage, all regional minima have a distance equal to MAXDIST, the initialization value since no smaller pixels are found in their neighborhood.

3.2 The lower-complete transformation

Before the lower-complete image is computed, the value of λ must be generated. Thus, each process P_i computes a local value $\lambda_i = \max\{d_i(p) \mid p \in D_{d_i}, d_i(p) < \text{MAXDIST}\} + 1$. These values are centralized by the master process to compute the global maximum $\lambda = \max_{1 \leq i \leq m \times n} \{\lambda_i\}$. λ is then broadcasted to all the processes P_i to map each subimage f_i to its lower-complete version l_i . Note that minima must be treated separately when applying the lower-complete transformation since their lower distance is MAXDIST, and not 0, as in the definition. Therefore, if $d_i(p) = \text{MAXDIST}$, $l_i(p) = \lambda \times f_i(p)$, else $l_i(p) = \lambda \times f_i(p) + d_i(p)$. The lower-complete image retains the order of pixels which have different graylevels in the original image, and introduces a new ordering relation between pixels having the same graylevel, but different lower distances. Thus, a 2D ordering relation (graylevel in one dimension and lower distance in another dimension) is reduced to 1D—the lower-complete value.

While the lower-complete subimages l_i are gen-

erated, the integration constraints f_i are initialized and the output subimages o_i updated. At the initialization time, all the pixels within o_i were set to NARM(Not A Regional Minima). But pixels in the boundary of each subimage were marked with a special label IMOUTBORD. Since minima became distinguishable from other pixels ($d_i(p) = \text{MAXDIST}$), the following settings are done: $\forall p \in D_{l_i}, f_i(p) \leftarrow l_i(p), o_i(p) \leftarrow \text{INIT}$ (a special initialization label) for minima, and $f_i(p) \leftarrow \text{MAXDIST}$, otherwise.

Henceforth, the original image is replaced by its lower-complete version, and the gradient of the latter is computed in each subimage ∂_i^d according to the definition. Since the lower-complete mapping is a point operation, and l_i subimages are overlapping, no communication between processes is needed neither for the lower-complete transformation nor for the gradient computation.

3.3 Labeling the minima pixels

This stage consists of labeling each regional minimum (labeled with INIT above) with a different label. Labels run from 1 up to a maximum value denoted by MAX_LABEL. The values of the labels are ordered such that $\text{MAX_LABEL} < \text{INIT} < \text{NARM} < \text{IMOUTBORD}$. Each process P_i uses an equally and sufficiently long subrange of labels of $\text{length} = \frac{\text{MAX_LABEL}}{m \times n}$, starting at $\text{label} \leftarrow (i - 1) \times \text{length} + 1$.

The labeling is performed in a very simple fashion, but with several scannings. First, each process iterates through a loop to label regional minima inside its distribution subdomain $D_{o_i^d}$. An iteration is the following:

```

(1) changed ← FALSE
(2) Raster_scan( $p \in D_{o_i^d}$ ) {
    (2.1) for each  $p' \in N_G^-(p)$ 
        if ( $o_i(p) \neq \text{NARM}$  and  $o_i(p') < o_i(p)$ ) {
             $o_i(p) \leftarrow o_i(p')$ ; changed ← TRUE
        }
    (2.2) if ( $o_i(p) = \text{INIT}$ ) {
         $o_i(p) \leftarrow \text{label}$ ; label++; changed ← TRUE
    }
}
(3) Anti-raster_scan( $p \in D_{o_i^d}$ ) {
    (3.1) do step 2.1
}

```

This first loop continues until *changed* stabilizes at FALSE. Then, parts of regional minima in different subimages must be merged since they received different labels in different processes. Therefore, each process P_i sends to all its neighbors $P_j \in N_G(P_i)$ the labels in the edge $E(o_i, o_j)$, and receives in exchange labels to be stored in the

boundary $B(o_i, o_j)$. A similar loop with that described above, but without step (2.2), is initiated to update labels in the distribution domain based on the new received ones in the extension area. New changes in the edges are communicated back to neighboring processes. This loop consisting of update and communication terminates when no changes have been registered in any process. It is again the role of the master process to detect as in the first stage when the computation stabilizes.

3.4 Flooding by image integration

As explained above, a catchment basin associated with a regional minimum is obtained by assigning the label of the latter to all pixels lying on minimal paths originating from that minimum. This is also computed by repeated raster and antiraster scannings in each distribution subdomain. A pixel p gets a label from a pixel $p' \in N_G^-(p)$ if $f_i(p) > f_i(p') + \partial_i(p)$, and sets $f_i(p)$ to $f_i(p') + \partial_i(p)$. Scannings are repeated as long as changes occur. During this raster and antiraster scannings, pixels within the edges update their labels and integration values based on labels and integration values of pixels in the boundaries. The latter locations in a subimage are always accessed for reading only. They are written by the corresponding neighboring processes, being parts of those particular distribution subdomains. Therefore, even if the computation stabilized in a process, new integration values and labels may have been set in neighboring processes for the boundary pixels. In order to compute the shortest paths within the global image, replica of the edges of the integration subimage f_i and labels o_i within each process P_i are done by message passing in the paired boundaries of each corresponding neighboring subimages f_j and o_j , respectively ($B(o_j, o_i) \leftarrow E(o_i, o_j)$, $B(f_j, f_i) \leftarrow E(f_i, f_j)$). New raster and antiraster scans are initiated to update labels and integration values based on the new data and changes are communicated back to neighboring processes. Scannings and communications are repeated until no changes occur in all subimages. This moment is detected by the master process.

4 Simulations

The algorithm described above was integrated into the parallel image processing system PIPS [5] on a massive MIMD parallel computer of type Parsytec SuperCluster 128 with distributed memory and reconfigurable network of transputers. These are T805 transputers with 4 Mbytes RAM

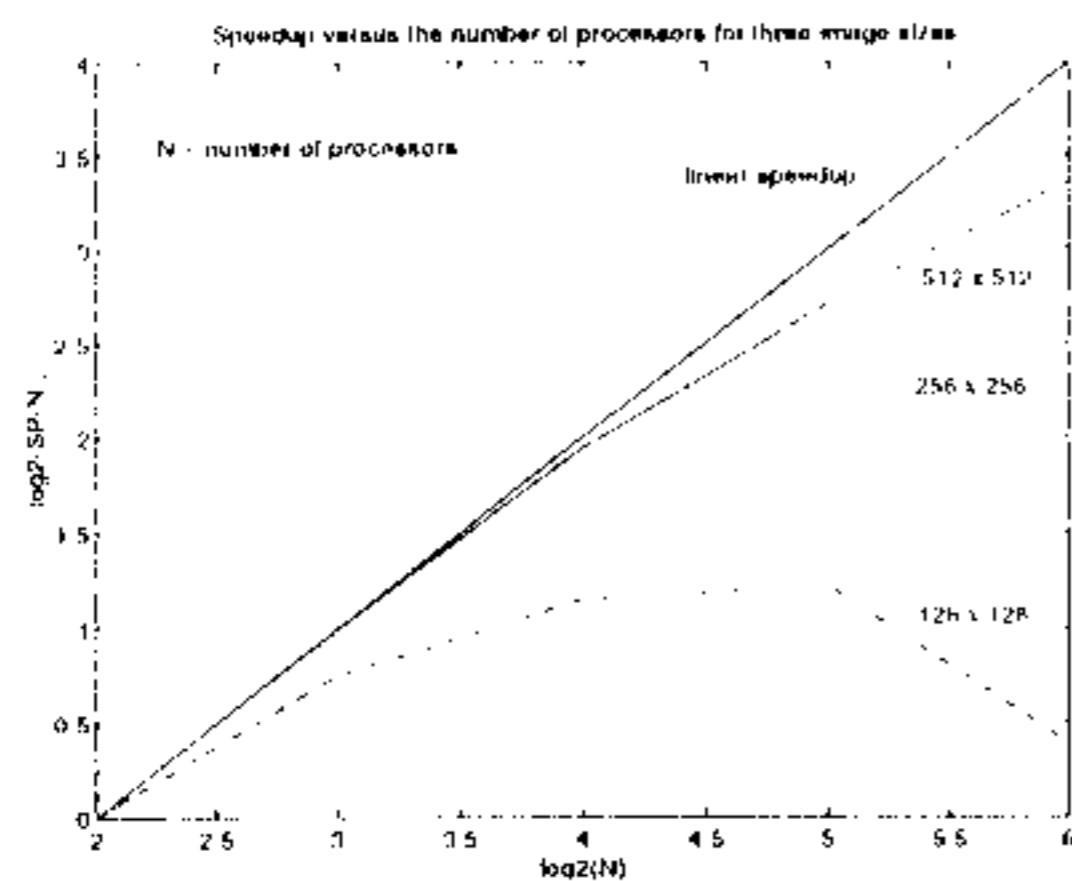


Figure 1: The performance of the Sequential-Scanning based Parallel Watershed Algorithm

and 4 links each. The links are bidirectional serial channels which perform a point-to-point communication with neighboring transputers at a rate of 1.8 Mbytes/second.

For the images we tested, the proposed watershed algorithm utilizes increasing processing resources efficiently. Thus, it is scalable up to a certain number of processors. The running time $T(N)$ is the time elapsed between the moment the first processor starts computing and the moment the last processor ends computing when N processors are used. The relative speedup is defined as $Sp(N) = T(N_{min})/T(N)$, where N_{min} is the minimum number of processors that the algorithm can handle, and $N > N_{min}$ is the number of processors for which the speedup is computed. Here, $N_{min} = 4$ since for large images the required amount of memory for one process exceeds the available memory per transputer. The running time and speedup of watershed algorithms are, of course, data dependent. In Figure 1, we plot the average speedup for different images of three different sizes as a function of the number of processors used. From the figure, the following can be observed:

- the speedup increases with the number of processors, but not linearly;
- the speedup tends to become saturated and the curve flattens;
- a larger size image yields higher speedup for the same number of processors.

One of the images we tested is part of the National Library of Medicine's Visible Human Program. The original image is shown in Figure 2. Figure 3 illustrates the output of the watershed algorithm when the input is the morphological gradient of the image in Figure 2, thresholded with a value of 20.

References

[1] S. Beucher and F. Meyer, "The morphological approach to segmentation: The watershed transformation," in *Mathematical Morphology in Image Processing*, E. R. Dougherty, Editor, pp. 433-481, Marcel Dekker Inc., New York, 1993.

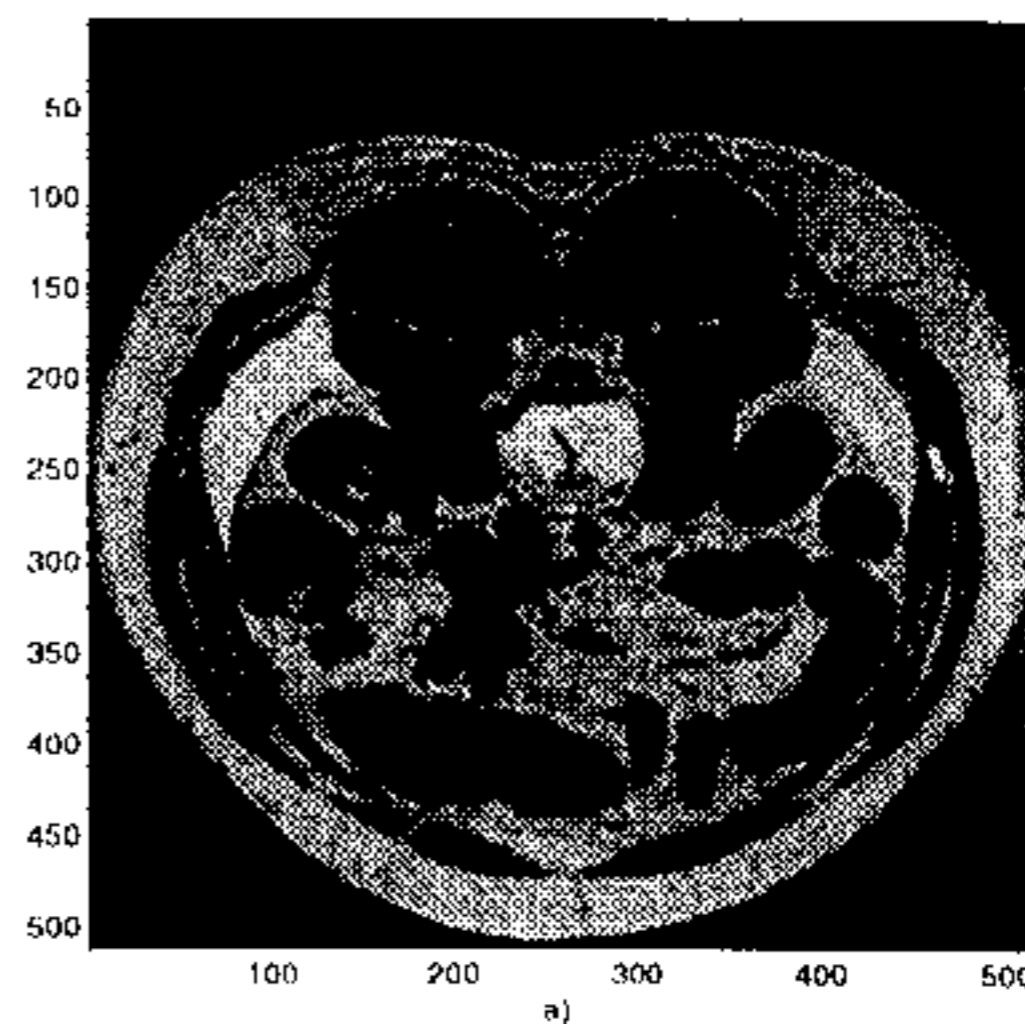


Figure 2: Input image: *Body cross-section*

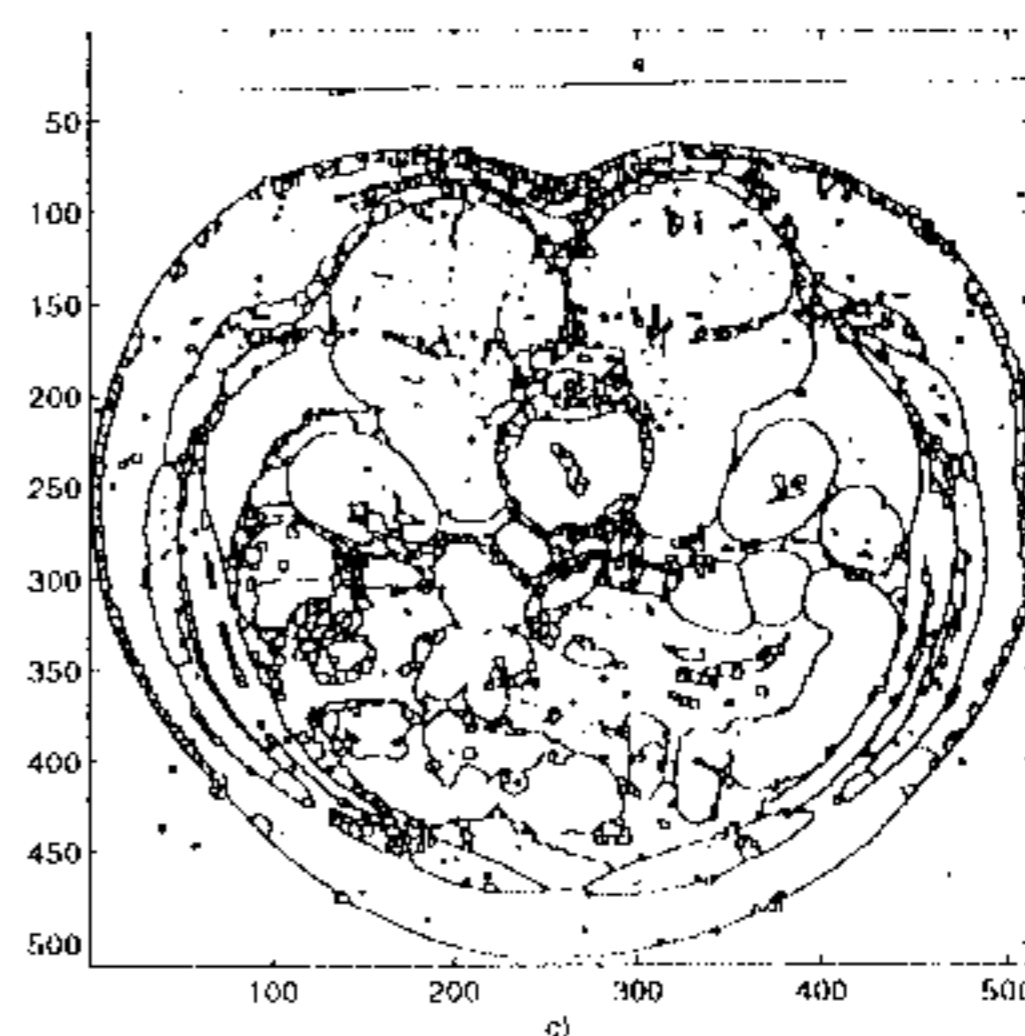


Figure 3: Output image: *Body cross-section*

- [2] F. Meyer, "Integrals and Gradients of Images," in *Proc. SPIE, Image Algebra and Morphological Image Processing III*, pp. 200-211, San Diego, 1992.
- [3] F. Meyer, "Integrals, gradients, and watershed lines," in *Proc. Workshop on Mathematical Morphology and its Applications to Signal Processing*, pp. 70-75, Barcelona, Spain, May 1993.
- [4] A.N. Moga, T. Viero, B.P. Dobrin, and M. Gabbouj, "Implementation of a Distributed Watershed Algorithm," in *ISMM'94 Mathematical Morphology and Its Applications to Image Processing*, pp. 281-288, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994.
- [5] M. Nölle, G. Schreiber, and H. Schulz-Mirbach, "PIPS a general purpose Parallel Image Processing System," in *16. DAGM - Symposium "Mustererkennung"*, G. Kropatsch Editor, Wien, September 1994. Reihe Informatik XPress, Springer.
- [6] A. Rosenfeld and J.L. Pfaltz, "Sequential Operations in Digital Picture Processing," in *Journal of the ACM*, vol. 13, no. 4, pp. 471-494, Oct. 1966.
- [7] P.W. Verbeek and B.J.H. Verwer, "Shading from Shape, the Eikonal Equation Solved by Gray-Weighted Distance Transform," in *Pattern Recognition Letters*, vol. 11, pp. 681-690, Oct. 1990.