Topological Numbering of Features on a Mesh

Mikhial J. Atallah
Susanne E. Hambrusch
Lynn E. TeWinkel

September 1988

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 88.25

NASA Cooperative Agreement Number NCC 2-387

RIACS

Research Institute for Advanced Computer Science

Topological Numbering of Features on a Mesh

Mikhial J. Atallah†
Susanne E. Hambrusch††
Lynn E. TeWinkel†††

Department of Computer Science
Purdue University
West Lafayette, IN 47907, USA

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 88.25
September 1988

Assume we are given a $n \times n$ binary image containing horizontally convex features; i.e., for each feature, each of its row's pixels form an interval on that row. In this paper we consider the problem of assigning topological numbers to such features; i.e., assign a number to every feature f so that all features to the left of f ub the image have a smaller number assigned to them. This problem arises in solutions to the stereo matching problem. We present a parallel algorithm to solve the topological numbering problem in $O(n)$ time on an $n \times n$ mesh of processors. The key idea of our solution is to create a tree from which the topological numbers can be obtained even though the tree does not uniqiely represent the "to the left of" relationship of the features.

---

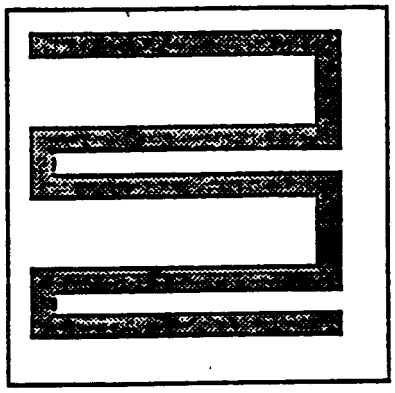
†Work reported herein was supported in part by Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration (NASA) and the Universities Space Research Association (USRA) while the author was a Visiting Scientist at the Center for Advanced Architectures at RIACS, and by the Office of Naval Research under Grants N0014-84-K-0502 and N00014-86-K-0689, and the National Science Foundation under Grant DCR-8451393, with matching funds from AT&T.

††This work was supported by the Office of Naval Research under Contract N0014-84-K-0502 and N00014-86-K-0689, and by the National Science Foundation under Grant MIP-87-15652.

†††This work was supported by the Office of Naval Research under Contract N00014-86-K-0689.

I. Introduction

Assume we are given an $n \times n$ binary image in which we refer to the image positions containing a '1' (resp. '0') as containing a 1-pixel (resp. 0-pixel). The connected components, which we call *features*, formed by the 1-pixels are *horizontally convex*; i.e., for each feature, each of its row's 1-pixels are contiguous on that row or, equivalently, form an interval on that row. An example of an image containing a horizontally convex feature is shown in Figure 1.



A horizontally convex feature

Figure 1

Observe that the features do not need to be vertically convex. For horizontally convex features the relationship "to the left of" defines a partial order as follows. A feature f is *to the left* of a feature g, denoted $f \{ g$, if and only if a 1-pixel of f is to the left of a 1-pixel of g on some row of the image. If no such row exists then f and g are *incomparable*. The *topological numbering problem* is to assign to every feature in the image a unique number so that for any feature f, all features to the left of f have a smaller number assigned to them. The need for topological numbering of features arises in solutions to the stereo matching problem [OK]. In this paper we present a parallel algorithm that determines the topological numbers in $O(n)$ time when the image is stored in an $n \times n$ mesh of processors, one pixel per processor.

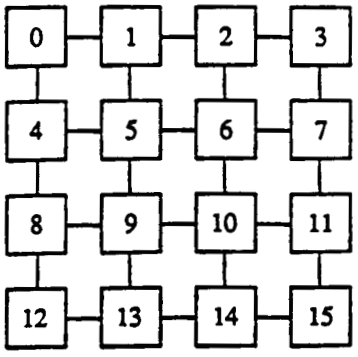
We next explain briefly how the topological numbering problem arises in the context of the stereo matching problem. In the stereo matching problem we are given two images, L and R, of a 3D scene and the objective is to determine the coordinates (x,y,z) of a point in the 3D scene from its image points (x_l,y_l) in L and (x_r,y_r) in R. The heart of the stereo matching problem is finding corresponding points in the two images of the same scene [BF, BT, GK, MP]. An

elegant dynamic programming solution is given by Ohta and Kanade [OK]. One of the subproblems that needs to be solved in their algorithm is that of establishing precedence between horizontally convex features of an image by assigning topological numbers to them. The sequential algorithm given in [OK] for numbering features is based on performing a topological search on a directed acyclic graph. The directed acyclic graph is obtained by making every feature a vertex and introducing a directed edge from f to g if $f \nmid g$ and there exists at least one horizontal line in which no other feature is met when going from f to g on that line. The so obtained directed acyclic digraph has at most $O(n^2)$ vertices and $O(n^2)$ edges and a topological search on it correctly numbers the features. If topological numbers are obtained for both images L and R, then they are used to enforce consistency in the matching process as follows. If the algorithm decides to match feature f_l of L with feature f_r of R, then all features in L with a smaller number than f_l can only be matched with features in R that have a smaller number than f_r.

In the next section we describe our algorithm that determines the topological numbers in $O(n)$ time when the $n \times n$ image is stored in an $n \times n$ mesh of processors with one pixel per processor. We assume the mesh to be an SIMD (Single Instruction, Multiple Data) machine in which every processor has a fixed (i.e. $O(1)$) amount of of local storage [B]. Our algorithm determines the topological numbers by solving a graph problem. Its basic idea is quite different from the one used by Ohta and Kanade [OK]. We cannot afford to parallelize the topological sort on that directed acyclic graph since topological sorting on such a graph does not seem to lend itself to an efficient parallel solution. Instead, in our solution we construct from the image a rooted, directed in-tree and we obtain the topological numbers by performing computations on that tree.

II. The Algorithm

Throughout we will index the processing elements (PEs) of the mesh by either their row-major index or by their row/column position. Figure 2 illustrates the row-major indexing scheme. The PE at row i and column j is denoted by PE(i,j). We let position (0,0) of the mesh (and thus the image) be the top left PE of the mesh. The input to our algorithm is an $n \times n$ binary image I, where $I(i,j)$ is stored at PE(i,j). Our algorithm assumes that every PE containing a 1-



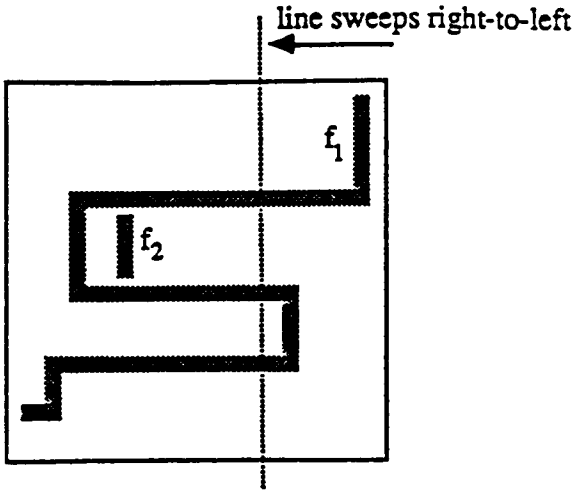
A 4×4 mesh with row-major indexing

Figure 2

pixel also contains the connected component number of the corresponding feature. We furthermore assume that the component numbers are assigned so that the number of a feature is the largest row-major index of any PE containing a 1-pixel from that feature. That is, the number of a feature is the row-major index of the lowest rightmost 1-pixel in it. Numerous parallel mesh algorithms to label the connected components in $O(n)$ time are known [CSS, H, HT, MS, NS] and they can easily be modified to label the features as needed by our algorithm. We refer to the component number stored at PE(i,j) as $C(i,j)$. The output of our algorithm is array TN, where $TN(i,j)$ contains the topological number of the feature containing the 1-pixel at position (i,j).

One seemingly obvious algorithm for determining the topological numbers is based on the "sweepline" method. A horizontal line is used to sweep the image, for example, from right to left. The first time a 1-pixel of a feature g is encountered, it is given a topological number such that feature g is given a larger number than feature f if and only if feature g is encountered before feature f in the right-to-left sweep. An image on which the method fails is shown in Figure 3.

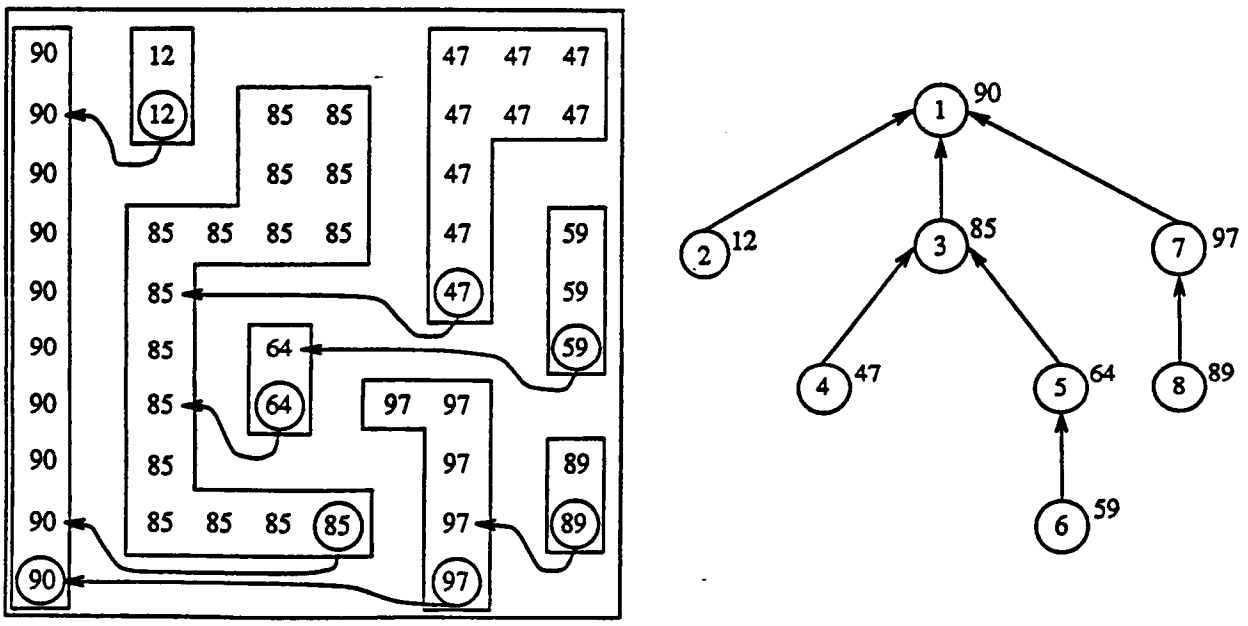
Our algorithm consists of three steps which are described next. In the first step it creates from the input image I a rooted, directed in-tree T in which the features correspond to the vertices. We create a root vertex which corresponds to an artificially created feature running along the length of the left border of the input image (introducing this "dummy" feature simplifies the exposition). The edges of the tree are created as follows. If PE(i,j) contains a component number equal to its row-major index, then PE(i,j) initiates a horizontal scan leftward on row i.



Illustrating why a "sweepline" method does not work

Figure 3

The component number is taken along on this scan. In Figure 4(a), the processors initiating such a scan are circled. Thus a processor initiates a scan if and only if it contains the lowest right-most 1-pixel of a feature.



(a) Features indicated by component numbers; arrows show scans done in step 1

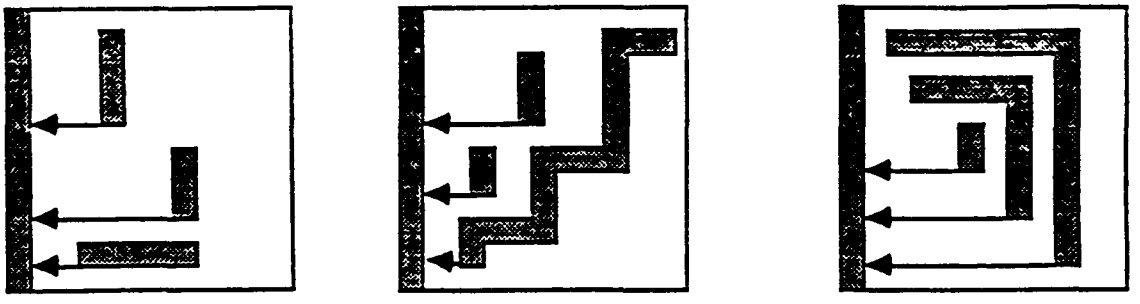
(b) Rooted directed tree created from (a) with preorder and component numbers

Figure 4

From now on we will refer to the largest row number (i.e. geometrically lowest row) containing 1-pixels belonging to feature f as low(f). For example, in Figure 4(a), low(12)=2, and

low(47)=5. If feature f contains more than one 1-pixel in low(f), then the scan may travel through other 1-pixels belonging to f. The scan initiated at PE(i,j) stops when it reaches the first 1-pixel belonging to a different feature g. For example, in Figure 4(a), the scan initiated at the circled '85' stops at the tip of the arrow starting from it (i.e., at a '90'). If the scan stops at PE(i,j'), $j'<j$, the directed edge ($C(i,j)$, $C(i,j')$) is created in the PE(i,j'). The graph created in this step is an acyclic digraph in which every node has out-degree exactly one, except one node which has out-degree zero (the "dummy" feature, labeled '90' in Figure 4(a)). It is easy to see that this graph is actually a directed in-tree rooted at the node of out-degree zero. Figure 4(b) shows the tree created from 4(a). In figure 4(b), notice that the children of any node are drawn from left to right by increasing component numbers. Thus T is, in some sense, an ordered tree.

The rooted, directed in-tree created by our algorithm does not contain all the information about the partial order. Figure 5 illustrates three distinct partial orderings that give rise to the same tree.



(a) (b) (c)

Illustrating distinct partial orderings that give rise to the same tree

Figure 5

Contrast this with the directed acyclic graph of Ohta and Kanade [OK], whose transitive closure is exactly the "$\prec$" relationship. However, the tree created in our algorithm contains all of the crucial information needed to compute the topological numbers, and the simple structure of a tree allows us to generate these numbers in $O(n)$ time.

The second step of the algorithm takes the rooted, directed in-tree T and computes the preorder numbers of the vertices of T using the Atallah-Hambrusch algorithm given in [AH]. When computing the preorder numbers it is crucial that the children of every node are visited in the order of increasing component numbers. As we will show in the next section, for any two

features f and h with with $f \{ h$, the preorder number of f is smaller than the preorder number of h. Hence, the preorder numbers are correct topological numbers. The preorder numbers for the tree shown in Figure 4(b) are given inside the circles indicating the vertices. After the preorder numbers have been assigned, the final step of the algorithm broadcasts them from the vertices of T to all 1-pixels of the corresponding features in I, thus creating the array TN. This "broadcasting" is done in $O(n)$ time using methods described in [H, MS, NS].

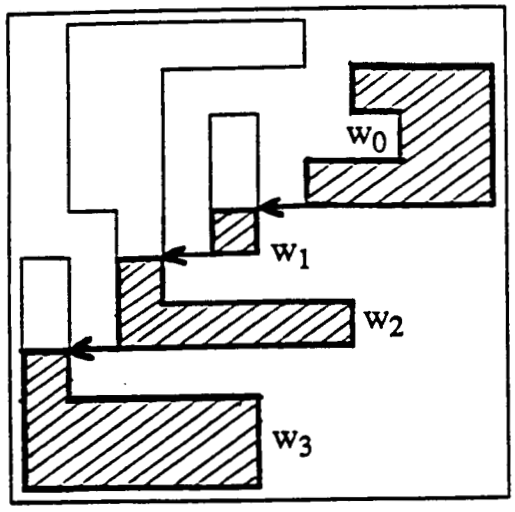
III. Correctness

In this section we prove the correctness of the algorithm outlined in the previous section. We denote the preorder number assigned to feature f as pre(f). Recall that in the algorithm preorder numbers are assigned to the vertices such that if a vertex has two children u and v and $u < v$, then pre(u) < pre(v) (i.e., vertex u is visited before vertex v in the traversal). Before showing that the preorder numbers are topological numbers, we introduce the notion of an image path, and then give a lemma.

Let $w_0, w_1, \cdots, w_{k-1}, w_k$ be the vertices on the path from feature w_0 to w_k in T. The *image path* $P(w_0,w_k)$ consists of all the 1-pixels in feature w_0, all the 1-pixels in feature w_i that are on a row greater or equal to low(w_{i-1}), $1 \le i \le k$, and of all the 0-pixels traversed during the horizontal scan from feature w_i to w_{i+1}, $0 \le i \le k-1$. See Figure 6 for an example of an image path $P(w_0,w_3)$. Observe that the set of pixels included in an image path is connected and horizontally convex.

Lemma. Let $P(w_0,w_k)$ and $P(u_0,u_l)$ be two image paths with $w_0 \neq u_0$ and $w_k \neq u_l$. There cannot exist a 1-pixel that is in both $P(w_0,w_k)$ and $P(u_0,u_l)$.

Proof. Assume there exists such a 1-pixel. If there exists more than one, choose the highest, rightmost one. Let x be this 1-pixel. An image path consists of portions of features and horizontal lines, and since x is in both image paths, x must come from a feature. W.l.o.g. let w_i be this feature. Both image paths contain all the 1-pixels of feature w_i within and below the row containing pixel x. Hence, both image paths contain the same set of 1-pixels from this point on and $w_k \neq u_l$ is not possible. □



Example of an image path $P(w_0, w_3)$; image path includes shaded portions of the features and the horizontal lines.

Figure 6

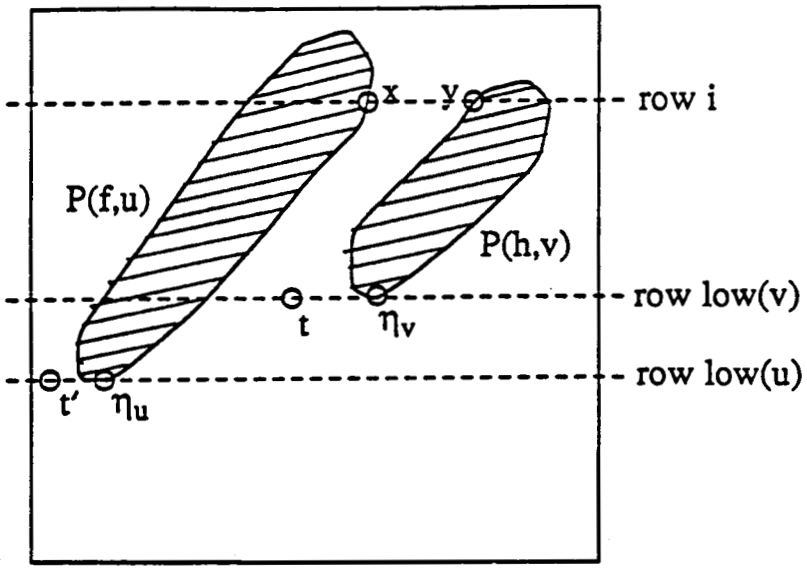
Theorem. For any two features f and h with $f \{ h$, we have pre(f) < pre(h).

Proof. Let f and h be any two features with $f \{ h$. If h is a descendant of f in T, then, due to the definition of preorder numbering, feature f is assigned a smaller preorder number than feature h and the theorem follows.

Assume now that h is not a descendant of f in T. Let d be the common ancestor of f and h that is lowest in T (i.e., farthest from the root in T). Let u and v be the two children of d such that f is a descendent of u and h is a descendent of v. To show that pre(f) < pre(h), it suffices to show that low(u) < low(v), since this would imply that u occurs before v in the ordered list of d's children (i.e., it would imply $u < v$). This in turn would imply that the preorder traversal visits u and its subtree before v and its subtree, and hence that pre(f) < pre(h). We now show that low(u) < low(v).

Assume by contradiction that low(u) ≥ low(v). If low(u) = low(v), then u and v cannot both be children of d in T, and therefore low(u) > low(v). Since $f \{ h$, there exists a row i such that a 1-pixel x of f is to the left of a 1-pixel y of h. Let η_u (resp. η_v) be the rightmost 1-pixel of u (resp. v) in row low(u) (resp. row low(v)). Figure 7 shows x, y, η_u, η_v, and the relative positions of the image paths $P(f,u)$ and $P(h,v)$. In particular, $P(f,u)$ contains the pixels x and η_u, $P(h,v)$ contains the pixels y and η_v, and η_v is to the right of $P(f,u)$. (If η_v was to the left

of $P(f,u)$, then $P(f,u)$ and $P(h,v)$ would intersect, violating the previous lemma).



Example of two image paths, $P(f,u)$ and $P(h,v)$; pixels x, y, η_u, η_v, t, and t' are circled and labeled.

Figure 7

Since d is the parent of v in T, there is a 1-pixel of d in between $P(f,u)$ and η_v on row low(v). Let t be such a 1-pixel. Since d is also the parent of u, there is a 1-pixel of d to the left of $P(f,u)$ on row low(u). Let t' be such a pixel. Figure 7 shows both t and t'. Since d is horizontally convex, the only way for t and t' to be in the same feature d is for d to cross $P(f,u)$. This is not possible and thus we have low(u) < low(v). □

IV. Implementation

We now show that each step of the algorithm can be implemented in $O(n)$ time on a mesh of size $n \times n$. All of the leftward scans are obviously completed within n time steps. It is also easy to see that at most one edge $(C(i,j), C(i,j'))$ is created in PE(i,j'). This holds since, for a 1-pixel located at position PE(i,j') belonging to a feature f, there is at most one PE(i,j) containing a 1-pixel belonging to a different feature g immediately to the right of PE(i,j') on row i. Once the rooted, directed in-tree T has been created, we use the algorithm described in [AH] to assign the preorder numbers to the vertices of the tree in $O(n)$ time. The algorithm in [AH] assigns preorder numbers to the vertices of T in the manner required by our algorithm; i.e., for

any vertex in T with children u and v, u is visited before v if and only if low(u) < low(v). The input to the preorder algorithm consists of the edges of T as stored in the PEs after the leftward scans have been completed. If edge $(C(i,j), C(i,j'))$ is stored in PE(i,j') before the preorder numbering occurs, then we can assume that the preorder algorithm returns $(C(i,j), C(i,j'))$ to PE(i,j'). In addition, we can assume that the preorder algorithm returns to PE(i,j') the topological number for the feature that contains the 1-pixel at PE(i,j).

The final step is to broadcast the preorder number stored in PE(i,j') to all 1-pixels of the corresponding feature. The broadcasting can be done in $O(n)$ time in a number of ways. One way is to first sort by component number all 1-pixels belonging to each feature f along with $C(i,j)$ and the preorder number stored in PE(i,j'). The correct topological number is then assigned to all the 1-pixels of f and the 1-pixels are then sorted back to their original position in image I. Another way of implementing the final step without an explicit sort first sends the preorder number from PE(i,j') rightward to PE(i,j). The array TN is then created by performing a connected component computation which accomplishes the broadcast. This concludes the description of our $O(n)$ time algorithm for determining the topological numbers of features.

V. Conclusion

In this paper we gave an $O(n)$ time algorithm for topologically numbering horizontally convex features of an $n \times n$ binary image on an $n \times n$ mesh of processors. The topological numbering problem is to assign to every feature in the image a unique number so that all features to the left of a feature f have a smaller number assigned to them. Our algorithm solves this problem by constructing a rooted, directed in-tree from the image and by determining the preorder numbers of the constructed tree.

References

[AH] M.J. Atallah and S.E. Hambrusch, "Solving Tree Problems on a Mesh-Connected Processor Array", *Information and Control*, Vol. 69, Nos. 1-3, pp. 168-187, April/May/June 1986.

[B] K.E. Batcher, "Design of a Massively Parallel Processor", *IEEE Transactions on Computers*, Vol. C-29, No. 9, pp. 836-840, September 1980.

[BF] S.T. Barnard and M.A. Fischler, "Computational Stereo", *Computing Surveys*, Vol. 14, No. 4, pp. 553-572, December 1982.

[BT] S.T. Barnard and W.B. Thompson, "Disparity Analysis of Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, No. 4, pp. 333-340, July 1980.

[CSS] R.E. Cypher, J.L.C. Sanz, and L. Snyder, "Practical Algorithms for Image Component Labeling on SIMD Mesh Connected Computers", *Proceedings of 1987 Int. Conference on Parallel Processing*, pp. 772-779, 1987.

[GK] C. Guerra and T. Kanade, "A Systolic Algorithm for Stereo Matching", in *VLSI: Algorithms and Architectures*, P. Bertolazzi and F. Luccio eds., North-Holland, pp. 103-112, 1985.

[H] S.E. Hambrusch, "VLSI Algorithms for the Connected Component Problem", *SIAM J. Comput.*, Vol. 12, No. 2, pp. 354-365, May 1983.

[HT] S.E. Hambrusch and L.E. TeWinkel, "A Study of Connected Component Labeling Algorithms on the MPP", *Proceedings of Third International Conference on Supercomputing*, Vol. I, pp. 477-483, May 1988.

[MP] D. Marr and T. Poggio, "Cooperative Computation of Stereo Disparity", *Science*, Vol. 194, No. 4262, pp. 283-287, 15 October 1976.

[MS] R. Miller and Q.F. Stout, *Parallel Algorithms for Regular Architectures*, manuscript, 1988 (to be published by MIT Press).

[NS] D. Nassimi and S. Sahni, "Finding Connected Components and Connected Ones on a Mesh-Connected Parallel Computer", *SIAM J. Comput.*, Vol. 9, No. 4, pp. 744-757, November 1980.

[OK] Y. Ohta and T. Kanade, "Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol PAMI-7, No. 2, pp. 139-154, March 1985.