International Journal of Parallel Programming, Vol. 20, No. 6, 1991

Optical Clustering on a Mesh-Connected Computer[1]

Frank Dehne,[2] Russ Miller,[3] and Andrew Rau-Chaplin[4]

Received February 1992; Revised July 1992

In this paper, we present optimal parallel algorithms for optical clustering on a mesh-connected computer. *Optical clustering* is a clustering technique based on the principal of optical resolution, and is of particular interest in picture analysis. The algorithms we present are based on the application of parallel algorithms in computational geometry and graph theory. In particular, we show that given a set S of N points in the Euclidean plane, the following problems can be solved in optimal $O(\sqrt{N})$ time on a mesh-connected computer of size N.

1. Determine the optical clusters of S with respect to a given separation parameter.
2. Given an interval $[a, b]$ representing the number of optical clusters desired in the clustering of S, determine the range of the separation parameter that will result in such an optical clustering.

KEY WORDS: Mesh-connected computer; optical clustering; image processing; computational geometry; connected components.

1. INTRODUCTION

This paper is concerned with parallel solutions to two problems in the area of clustering on a mesh-connected computer. We study the parallelization of *optical clustering*,[1] which is a clustering technique based on the principal of optical resolution, and which is of particular interest in picture

[1] Research partially supported by the Natural Sciences and Engineering Research Council of Canada and the National Science Foundation under Grant IRI-9108288.

[2] School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6.

[3] Department of Computer Science, State University of New York at Buffalo, Buffalo, New York 14260.

[4] School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6.

analysis. The algorithms we present are asymptotically optimal for the mesh-connected computer.

The input to both problems is a set of N points in the plane, which usually represent the pixels of an image (see Fig. 1). For the mesh-connected computer, each processor initially stores one of the input points, with an arbitrary assignment of points to processors. For *optical clustering*,[1] a relation *connected* is defined as follows. Two points are *connected* if and only if there exists a circle with radius $\leqslant r$ containing the two points. Since the transitive closure of this relation is an equivalence relation, the *connected components* of a set of points are defined to be the equivalence classes generated by the transitive closure over the relation *connected*. The first clustering problem studied in this paper consists of computing the equivalence classes (i.e., determining the connected components) for a given parameter r. The second clustering problem, which is more difficult, is the inverse problem. As input we specify a *range* over the number of connected components we expect. The problem is to determine a maximal interval for the parameter r such that the number of components of the respective clusterings is within the specified range. Sequential $O(N \log N)$ time solutions to both problems have been presented in Ref. 1. In this paper, we present optimal $(O(\sqrt{N})$ time parallel algorithms for the mesh-connected computer. A direct parallelization of the algorithms presented in Ref. 1 would yield $O(\sqrt{N} \log N)$ time parallel solutions on the mesh-connected computer. The main contribution of this paper is a nontrivial data compression technique which reduces the time of the mesh-connected computer algorithm to $O(\sqrt{N})$, which is optimal. Our solutions to these clustering problems involve parallel techniques and algorithms from computational geometry and graph theory.

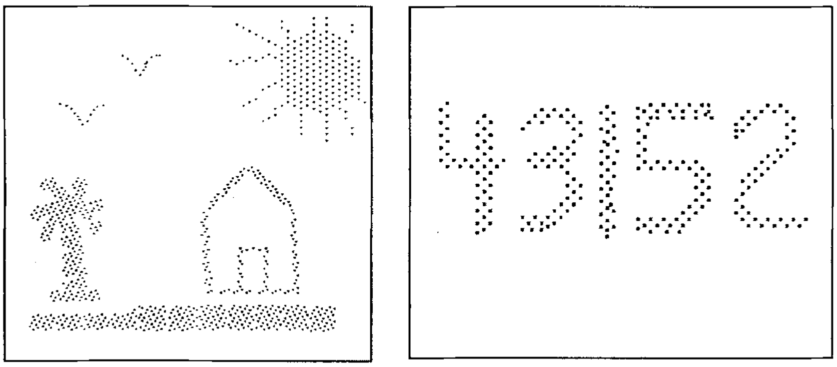

Fig. 1. Sample images. (a) relation *r-connected* for a set S'; (b) the graph $(S', \odot_r)$.

Our main motivation for studying parallel optical clustering on the mesh is that (1) the mesh-connected computer is a very popular architecture for image processing and (2) optical clustering, as part of a picture recognition system, often has on-line response time requirements, which can only be met by using parallel architectures or by creating customized hardware. Next, we outline particularly interesting applications of our techniques which have such on-line requirements. These problems were brought to our attention by Ed Cohen and Jon Hull from the CEDAR project at the University of Buffalo.

The second clustering problem outlined earlier has applications in pattern recognition systems where one knows a range in the number of objects that are anticipated, and would like to perform a clustering operation that will yield the anticipated number of objects. For example, such knowledge can be exploited when considering the digitization of a handwritten social security number, where one would anticipate approximately 11 distinct object (9 digits and two hyphens). In fact, since some of the handwritten digits in a social security number may overlap each other, while other individual digits might not even be connected, a suitable range in the number of clusters expected for a handwritten social security number might be, say, between 7 and 15. Once the objects are identified, there are numerous strategies, such as template matching or recognition based on feature sets, for attempting to recognize the digits. Another interesting application of optical clustering arises in the problem of determining the location of a handwritten address on an envelop.(2)

The remainder of this paper is organized as follows. In Section 2, the mesh-connected computer and the two problems considered in this paper are defined. In Section 3, we present optimal mesh solutions to both clustering problems. Section 4 outlines how these results can also be applied to images represented by chain codes, and Section 5 concludes the paper.

2. DEFINITIONS

In this section, we define the mesh-connected computer and the optical clustering problems considered in this paper.

2.1. The Mesh-Connected Computer

The *mesh-connected computer (mes) of size N* is an SIMD machine with N simple *processors* arranged in a square lattice. To simplify exposition, it is assumed that $N = 4^c$, for some integer c. For all $i, j \in [0,..., N^{1/2} - 1]$, let $P_{i,j}$ represent the processor in row i and column j.

Processor $P_{i,j}$ is connected via bidirectional unit-time communication links to its four *neighbors*, $P_{i-1,j}$, $P_{i+1,j}$, $P_{i,j-1}$, and $P_{i,j+1}$, assuming they exist. Each processor has a fixed number of $\Theta(\log N)$-bit words of memory (registers), and can perform standard arithmetic and Boolean operations on the contents of these registers in unit time. Each processor can also send or receive a word of data to or from each of its neighbors in unit time.

The communication diameter of a mesh of size N is $\Theta(\sqrt{N})$, as can be seen by examining the distance between processors in opposite corners of the mesh. This means that if a processor in one corner of the mesh needs data from a processor in another corner of the mesh at some time during an algorithm, then a lower bound on the running time of the algorithm is $\Omega(\sqrt{N})$. It is easy to see that, because of the communication diameter, the problems in this paper have time complexities $\Omega(\sqrt{N})$.

In this paper, we will frequently use $\Theta(\sqrt{N})$ time *standard mesh operations* such as sorting, random access read, random access write, compression, and parallel prefix. The reader is referred to Refs. 3–7, and the references contained therein, for complete descriptions, algorithms, and analyses of these operations.

2.2. Optical Clustering

This section gives a brief review of the definition and some basic properties of *optical clustering* as described in Ref. 1.

Let $S=\{s_1,..., s_N\}$ be a set of N disjoint objects in n-space, $\Re^n$ (i.e., compact subsets of $\Re^n$ without holes). Let d: $\Re^n \times \Re^n \rightarrow \Re^+$ be a convex distance function, where $\Re^+$ is the set of positive real numbers. For two objects $s, s' \in S$, define $d(s, s')$ as the minimum distance $d(x, x')$ between two points $x \in s$, $x' \in s'$. Finally, let $c(P, r) = \{x \in \Re^n \mid d(P, x) \leqslant r\}$ denote the ball with center $P \in \Re^n$ and radius r.

Consider two objects s_i, $s_j \in S$. We say that s_i and s_j are *r-connected*, denoted by $s_i \odot_r s_j$, if and only if there exists a ball $c(P, r')$, with $r' \leqslant r$, such that $c(P, r') \cap s_i \neq \emptyset$ and $c(P, r') \cap s_j \neq \emptyset$.

Since the transitive closure of the *r-connected* relation, denoted $cl(\odot_r)$, is an equivalence relation, we define the *optical clusters with respect to separation parameter* r as the equivalence classes of $cl(\odot_r)$.

Figure 2a illustrates the optical clusters of an 11 object set S' in $\Re^2$. Notice that the optical clustering of S' for the given value of r (illustrated by the balls of radius r) results in three clusters, namely, $\{s_1,..., s_5\}$, $\{s_6,..., s_{10}\}$ and $\{s_{11}\}$.

Let $m(S, r)$ denote the number of optical clusters of S with respect to r. Clearly, as r increases to infinity, the number of optical clusters $m(S, r)$

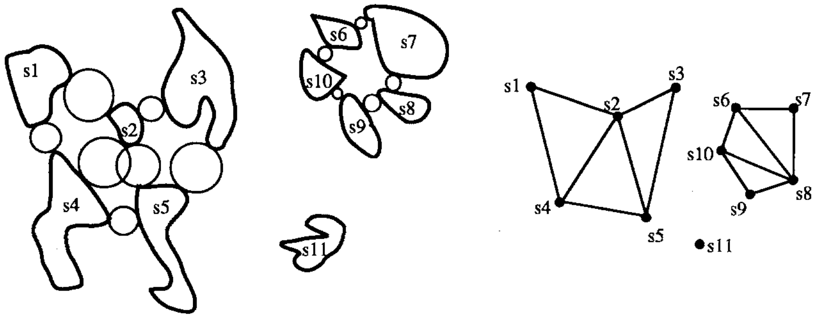

Fig. 2. Illustration of the relation *r-connected.* (a) relation *Delaunay connected* for a set S'; (b) the graph (S', Δ_r).

decreases to 1. In fact, $m(S, r)$ is a monotonically decreasing function in r. That is, $r \leqslant r' \Rightarrow m(S, r) \geqslant m(S, r')$.

Given the task of constructing the optical clusters of a set S of geometric objects with respect to a separation parameter r, a naive solution would be to compute the graph $(S, \odot_r)$ (see Fig. 2b), and then find the connected components of $(S, \odot_r)$. The drawback to this approach is that it involves computing the connected components of the graph $(S, \odot_r)$, which has size $O(N^2)$, since in the worst case it may be a complete graph on the N vertices. As an illustration of this, consider the optical clusters of the set $S'' = \{s_6, ..., s_{10}\}$ from Fig. 2a. Every object in S'' is *r-connected* to every other in S'' (witness the ball in the center of S''), implying that $(S'', \odot_r)$ has $|S''|^2$ edges. The $O(N^2)$ size of $(S, \odot_r)$ implies that the processor-time product of the naive algorithm is $\Omega(N^2)$ in the worst case.

Consider $s_i, s_j \in S$. We say that s_i and s_j are *Delaunay connected with respect to r*, denoted by $s_i \Delta_r s_j$, if and only if there exists an $r' \leqslant r$ and $P \in \mathfrak{R}^n$ such that $d(P, s_i) = r' = d(P, s_j)$ and for all $s_k \in S - \{s_i, s_j\}$, $d(P, s_k) > r'$.[1] Figure 3a illustrates the *Delaunay connected* relation with respect to the same set of objects S' given in Fig. 2a.

It has been shown in Fig. 3 that $cl(\Delta_r) = cl(\odot_r)$ and, for object sets in $\mathfrak{R}^2$, $|\triangle_r| = O(N)$. Hence, relation Δ_r induces the same clustering of S as relation $\odot_r$, but it has only a linear number of elements.

Since we are only interested in the equivalence classes of $cl(\odot_r)$, we can avoid the $\Omega(N^2)$ worst-case time-processor bound of the naive algorithm by using instead the relation Δ_r. The optical clusters of S with respect to separation parameter r are exactly the connected components of the graph (S, Δ_r), where (S, Δ_r) denotes the graph with vertex set S and edge set containing all edges between all pairs of vertices $(s, s') \in \Delta_r$. (See Fig. 3b.)

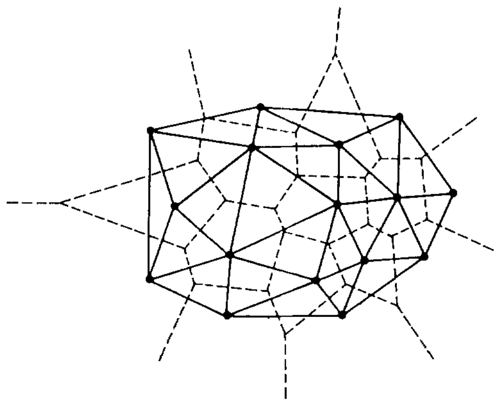

Fig. 3. Illustration of the relation *Delaunay conected.*

Let $V(S)$ and $DT(S)$ denote the *Voronoi Diagram* and its dual, the *Delaunay Triangulation*, of S, respectively Refs. 8 and 9; see Fig. 4. It is easy to see that $DT(S) = \bigcup_{r \geq 0} \Delta_r$. Define for every edge $(s, s') \in DT(S)$ with corresponding dual edge e in $V(S)$ a label

$$\min(s, s') = \min\{d(s, x) = d(s', x) \mid x \in e\}$$

and call the labeled graph $(S, DT(S))$, with laveling $(s, s') \rightarrow \min(s, s')$, the *cluster graph* of S, denoted $CG(S)$. As shown in Ref. 1, it follows that $\Delta_r = \bigcup_{[(s,s') \in DT(S), \min(s,s') \leq r]} (s, s')$.

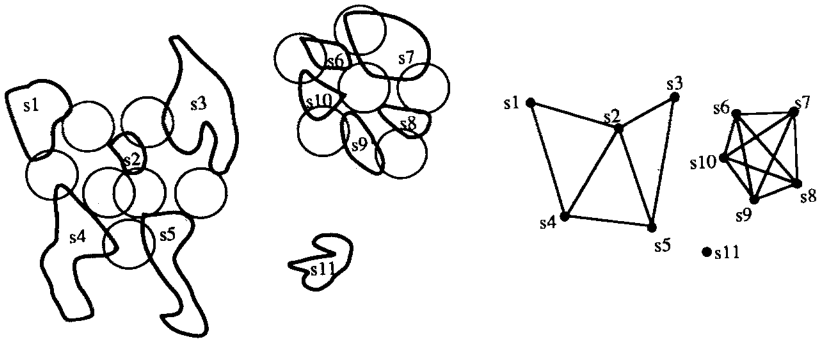

Fig. 4. A Voronoi diagram (solid lines) and Delaunay triangulation (dashed lines).

3. PARALLEL OPTICAL CLUSTERING OF POINTS SETS IN E^2

3.1. Computing Optical Clusters with Respect to a Given Separation Parameter r

Let $S = \{s_1, ..., s_N\}$ be a set of distinct planar points, arbitrarily distributed one per processor on a mesh of size N. In this section, we consider the problem of determining the optical clusters of S with respect to a given separation parameter r. The algorithm consists of first constructing $CG(S)$, and then computing the connected components of an edge-restricted subgraph of $CG(S)$.

In $O(\sqrt{N})$ time, we can construct the Voronoi diagram $V(S)$ and its dual, the Delaunay triangulation $DT(S)$, using the algorithm given in Ref. 10. At this stage, every processor stores one (arbitrary) edge of $DT(S)$ and the corresponding edge of $V(S)$. To complete the construction of $CG(S)$, we compute in $\Theta(1)$ time, simultaneously for every edge $(s, s') \in DT(S)$, its label $\min(s, s')$.

Lemma 1. The cluster graph $CG(S)$ of a set of N points in the Euclidean plane can be computed in optimal $O(\sqrt{N})$ time on a mesh of size N.

Given the cluster graph $CG(S)$ and a real value $r > 0$, we can compute the optical clusters with respect to separation parameter r as follows. Delete all edges $(s, s') \in CG(S)$ with label $\min(s, s') > r$, and compute the connected components of $CG(S)$ with respect to the remaining edges. Note that this operation gives us the optical clusters of S with respect to r and allows us to compute the number of optical clusters, $m(S, rt)$, by a parallel prefix operation. Since the connected components of a graph with $O(N)$ edges can be computed in $O(\sqrt{N})$ time on a mesh of size N,(11,12) we have the following.

Lemma 2. Given a set S of N points in the Euclidean plane, and a real number $r > 0$, the optical clusters of S with respect to separation parameter r can be computed in optimal $O(\sqrt{N})$ time on a mesh of size N. In addition, the number $m(S, r)$ can be computed in optimal $O(\sqrt{N})$ time.

3.2. Computing Optical Clusters with Respect to a Given Range in the Number of Clusters Desired

In this section, we consider the problem of determining the maximal interval for the separation parameter r such that the number of clusters of the respective optical clusterings of a planar point set S is within a desired range.

Consider Fig. 1. If we perform optical clustering (as described in

Section 3.1) with respect to an r value that is too small, each cluster will consist of exactly one point, providing us with little additional information about the structure of the image. On the other hand, if we perform optical clustering with respect to a value r that is too large, then all the points in the image will form a single cluster, again providing little additional information about the structure of the image. Clearly, knowledge of a suitable r implies considerable knowledge concerning the structure of the image. In some applications, whereas we may not know a "suitable" value for r, we may have knowledge with respect to the number of optical clusters we expect the image to consist of. For example, suppose one is to process the digitization of some preprinted from. The fields might include information such as a social security number, credit card number, date of birth, age, height, weight, income, and so forth. For such fields, we can exploit our knowledge with respect to an acceptable range of the number of clusters that is anticipated, as discussed in further detail in the Introduction.

We will now show how knowledge about the expected number of clusters can be used to compute a suitable range of values for the separation parameter r. Let $[a, b]$, $a \leqslant b$, $a, b \in \{1,..., N\}$, be an interval denoting the desired range of $m(S, r)$, the number of clusters of S with respect to r. Let $R(a, b) \subset \Re^+$ denote the corresponding set of separation parameters. That is, $r \in R(a, b)$ means that $m(S, r) \in [a, b]$. Let $R(a)$ be the largest value $r' \in \Re^+$ such that $m(S, r') \geqslant a$. Similarly, let $R(b)$ be the smallest value $r' \in \Re^+$ such that $m(S, r') \leqslant b$. Since $m(S, r)$ is monotone and decreasing with respect to r, $R(a, b)$ is the closed interval $[R(b), R(a)]$. Let $CG(S)$ have k edges, and let $\min_1,..., \min_k$ be the values of the labels of these edges in increasing sorted order. Thus, $m(S, \min_1) \geqslant m(S, \min_2) \geqslant \cdots \geqslant m(S, \min_k)$, and for all i, $1 \leqslant i < k$, if $r \in [\min_i, \min_{i+1})$, then $m(S, r) = m(S, \min_i)$. Given the definition of $CG(S)$, notice that $R(b)$ must be an element of $\{\min_1,..., \min_k\}$, and $R(a)$ must be a maximum end-point of some interval $[\min_i, \min_{i+1})$. Therefore, we can determine $R(a, b)$ by computing $R(a)$ and $R(b)$, where $R(a)$ and $R(b)$ can be computed independently.

We will use a binary search type algorithm to determine $R(a)$ and $R(b)$. Our algorithm will consist of $0 \leqslant t \leqslant \log_2 N$ phases, where in a phase, the original graph CG will be further *compressed*. A similar compression technique was previously used in Ref. 11. Let CG_t denote the compressed version of CG immediately following the tth stage of the algorithm. Let $CG_t^{r'}$ denote a copy of CG_t, where all edges with labels $m(S, s') \geqslant r'$ have been removed. Phase 0 of the algorithm is defined by (i) setting $CG_0 = G(S)$, where $CG(S)$ is constructed as in Section 3.1, and (ii) setting q_0 to either a or b, depending on which one is being searched for. The non-increasing sequence $\langle q_i \rangle = q_0,..., q_{\log N}$ is used to keep track of the number of optical clusters remaining to be identified following the termination of phase i. The tth phase of the algorithm is defined as follows.

1. Let $CG_{t-1} = (V, E)$ and $CG_t = (V', E')$, where CG_{t-1} is given and CG_t is the graph to be constructed in the tth phase. Determine r', the median over the set of edge labels of CG_{t-1}.
2. (a) Construct the graph $CG^{r'}_{t-1}$ by eliminating all edges with labels greater than or equal to r'.
   (b) Label the connected components of $CG^{r'}_{t-1}$.
   (c) Compute the number of connected components of $CG^{r'}_{t-1}$.
3. Terminating Cases: (T1) If the number of connected components of $CG^{r'}_{t-1} = q_{t-1}$, then return r'. EXIT. (T2) If $CG^{r'}_{t-1}$ consists of a single edge with lebel l, then return either the label just greater than l or just less than l in the sequence $\min_1, \ldots, \min_k$, depending on whether we are searching for $R(a)$ or $R(b)$, respectively. EXIT.
4. Case A: The number of connected components of $CG^{r'}_{t-1}$ is less than q_{t-1}, i.e., r' is too large. Set E' equal to the set of edges in $CG^{r'}_{t-1}$. Notice that any vertex $v \in V$ not incident on an edge in E' will remain as a separate cluster for the remaining phases of the algorithm, since in this binary search procedure the set of edges is never increased. Therefore, identify the set U of connected components of $CG^{r'}_{t-1}$ which contain exactly one vertex, and compute $|U|$, the number of single vertex components in $CG^{r'}_{t-1}$. Set V' equal to $V - U$ and $q_t = q_{t-1} - |U|$.
5. Case B: The number of connected components of $CG^{r'}_{t-1}$ is greater than q_{t-1}, i.e., r' is too small. This means that none of the edges in $CG^{r'}_{t-1}$ has a label that results in q_{t-1} clusters, and that every edge in $CG^{r'}_{t-1}$ links vertices that are members of the same optical cluster. Therefore, collapse the connected components of $CG^{r'}_{t-1}$ to form the set V' of "Super-Vertices," and let E' be the set of edges (s, s') between elements of V' such that there exists an edge e in CG_{t-1} connecting the components represented by s and s'. (Note: Remove all loops and collapse multiple edges in E.) Notice that, as is Case A, any singleton vertex in (V', E') will remain as a separate cluster for the remaining phases of the algorithm. Therefore, label the connected components of (V', E'), and identify the set W of connected components of (V', E') with exactly one vertex. Set $V' = V' - W$ and $q_t = q_{t-1} - |W|$.

For this algorithm, the correctness and time complexity depend on the relationship between the size of graphs CG_t and CG_{t-1}. We show that in each phase, the graph is compressed by at least a factor of 6/5.

Lemma 3. $|CG_t| \leqslant \frac{5}{6}|CG_{t-1}|$.

Proof. Let CG_{t-1} have e edges and v vertices, and let CG_t have e' edges and v' vertices. Since r' is the media over all edge labels in CG_{t-1}, the result of either Case A or Case B is that half of the edges of CG_{t-1} are removed. Therefore, $e' \leqslant e/2$. Further, since singleton vertices are removed from either the connected components of $CG_{t-1}^{r'}$ (Case A) or the connected components of (V', E') (Case B), we know that $v' \leqslant v$. Let $M = e + v$ be the size of CG_{t-1}. Since there are no isolated vertices in CG_{t-1}, we have $v \leqslant 2e$. Therefore, $M = e + v \leqslant 3e$. Finally, we obtain

$$\begin{aligned} M' &= e' + v' \leqslant e' + v \leqslant e/2 + v = e/2 + M - e = M - e/2 \\ &\leqslant M - M/6 = \tfrac{5}{6}M \end{aligned} \qquad \square$$

Given the preceding lemma, we can use standard mesh techniques (c.f., and the references contained therein) to compress the new graph constructed at the end of each phase into the upper-left $m \times m$ region of the mesh, where $m^2 \leqslant N/(6/5)^{t-1}$. Notice that standard mesh operations (c.f., Refs. 3, 4, and 13, and the references contained therein) such as sorting, broadcasting, component labeling, random access read, random access write, and parallel prefix can be used to complete the tth phase of the algorithm on a mesh of size N in $O(\sqrt{N/(\frac{6}{5})^{t-1}})$ time. Therefore, the running time of the algorithm is given by the recurrence $T(N) = T(\frac{5}{6}N) + O(\sqrt{N})$, which implies $T(N) = O(\sqrt{N})$.

Theorem 4. Given a set S of N points in the Euclidean plane and an interval $[a, b]$ for the number $m(S, r)$ of optical clusters desired in the completed clustering of S, the range of values for the separation parameter r that will result in such an optical clustering can be determined in asymptotically optimal $O(\sqrt{N})$ time on a mesh of size N.

4. OPTICAL CLUSTERING FOR CHAIN CODE REPRESENTATIONS

In practice, the input is often not given as an image but in its *chain code* representation. Chain codes are a classical approach in image processing for representing regions by border codes.(14,15) A chain code representation of a standard 8-connected component without holes is defined by the starting location (x and y coordinates) of a pixel on the border, followed by a sequence of directions indicating where the remaining border pixels are with respect to, say, the counterclockwise direction. Chain codes are used in many situations, such as handwriting analysis, where the image is relatively sparse. This not only saves space, but because the representation is often more compact, the algorithms that work on the chain codes are much more efficient than algorithms that would work on the raw image. Optical clustering methods can also easily be applied to chain codes without expanding the chain code to the (in general much larger) image. Suppose that an image is initially represented as a set of chain codes to total length c, where each 8-connected component, as well as all holes of the component, is represented by a labeled chain code. Further, suppose that each such labeled chain code is stored in a contiguous set of processors on a mesh olf size c (c.f., Ref. 14). In time $O(\sqrt{c})$, the chain codes can be transformed into a labeled set S representing the coordinates of the border pixels. Then, instead of solving the clustering problems for the chain codes, we simply apply our clustering methods to S, which yields the required result. Note that the size of s is still $O(c)$.

5. CONCLUSION

In this paper, we presented asymptotically optimal mesh-connected computer algorithms to solve two problems in optical clustering, both of which are important in certain areas of image analysis. To the best of our knowledge, these are the first parallel algorithms presented for these problems. mhese techniques allow one to determine on a mesh-connected computer (i) the optical clusters of a digitized picture for a given separation parameter r, and (ii) the maximal interval for the separation parameter r such that the number of clusters produced is within a certain range. This is important in applications such as on-line processing of handwriting (e.g., in the processing of handwritten forms) or determining the location of a handwritten address on an envelope.[2]

ACKNOWLEDGMENTS

We would like to thank Ed Cohen and Jon Hull of the CEDAR project at the University of Buffalo for several discussions.

REFERENCES

1. F. Dehne, Optical clustering, *The Visual Computer*, **2**:39–43 (1986).
2. P. W. Palumbo, S. N. Srihari, and J. Soh, Real-Time Address Block Location Using Pipelining and Multiprocessing, *IEEE Computer*, special issue on Document Image Analysis Systems, accepted for publication, to appear (June 1992).
3. F. T. Leighton, *Introduction to Parallel Algorithms and Architecturis: Arrays, Trees, Hypercubes*, Morgan Kaufmann Publishers, California (1992).

4. R. Miller and Q. F. Stout, *Parallel Algorithms for Regular Architectures*, manuscript to be published by MIT press.
5. D. Nassimi and S. Sahni, Bitonic Sort on a Mesh-Connected Parallel Computer, *IEEE Transactions on Computers*, **C-27**(1):2–7 (January 1979).
6. D. Nassimi and S. Sahni, Data Broadcasting in SIMD Computers, *IEEE Transactions on Computers*, **C-30**(2):101–107 (February 1981).
7. C. D. Thompson and H. T. Kung, Sorting on a Mesh-Connected Parallel Computer, *Communications of the ACM*, **20**(4):263–271 (April 1977).
8. M. I. Shamos and D. Hoey, Closest Point Problems, *SIAM J. on Comp.*, pp. 744–757 (1980).
9. P. Chew and R. L. Drysdale, Voronoi Diagrams Based on Convex Distance Functions, *Proc. of First Symposium on Computational Geometry* (1985).
10. C. Jeong and D. T. Lee, Parallel Geometric Algorithms on Mesh-Connected Computers, *Proc. of Fall Joint Computer Conference* (1987).
11. S. E. Hambrusch and F. Dehne, Determining Maximum k-Width Connectivity on Meshes, *Proc. Int. Parallel Proc. Symposium*, pp. 234–241 (1992).
12. J. Reif and Q. F. Stout, Optimal Component Labeling Algorithms for Mesh-Computers and VLSI (to appear).
13. D. Nassimi and S. Sahni, Finding Connected Components and Connected Ones on a Mesh-Connected Parallel Computer, *SIAM J. on Comp.*, pp. 744–757 (1980).
14. T. Dubitzki, A. Wu, and A. Rosenfeld, "Parallel Computation of Contour Properties, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **3**:331–337 (1981).
15. H. Freeman, Computer Processing of Line-Drawing Images, *Computing Surveys*, **6**:57–97 (1974).
16. R. Miller and Q. F. Stout, Mesh Computer Algorithms for Computational Geometry, *IEEE Transactions on Computers*, **38**(3):321–340 (March 1989).