In addition to an effective solution to the jigsaw puzzle problem, we have advanced the notion of a global feature of a planar object, called an isthmus. A method for reliably computing the isthmus feature from the Euclidean skeleton of an object and deriving a new set of critical points (isthmus critical points) that describe the feature has been presented.

Suggestions for Future Work

A set of programs that integrates many different critical points (sharp corners, inflection points, curvature maxima, and isthmus points) may broaden the class of puzzles solvable. Thus, if a solution is not possible using isthmus points, the algorithm could then use sharp corners or other points as the critical points.

The use of parallel processing to mate match segments could be used. It may be possible to store each puzzle piece's set of match segments in a small processor and then in one probe determine which of the other pieces best match. This could speed up the matching process. Future work might also include using a robot to physically assemble the puzzle. The robot end effector could be a suction or vacuum device to manipulate the pieces. The vision system could instruct the robot to perform the actual piece rotations and translations.

ACKNOWLEDGMENT

The authors wish to thank the IEEE reviewers for their very insightful suggestions, Dr. Giorgio Ingargiola and Dr. Charles Kapps of the Department of Computer Science at Temple University, Dr. Paul Ross of Millersville University and Dr. Marvin Ziskin of the Temple University School of Medicine and for their comments on earlier drafts of this paper.

REFERENCES

[1] H. Freeman, "Boundary encoding and processing," in *Picture Processing and Psychopictorics*, B. Lipkin and A. Rosenfeld, Eds. New York: Academic, 1970, pp. 101–120.

[2] G. Radack and N. Badler, "Jigsaw puzzle matching using a boundary-centered polar encoding," *Computer Graphics, and Image Processing*, vol. 19, pp. 1–2, May, 1982.

[3] R. Webster, "Partial boundary matching and shape fitting using the medial axis transformation," Ph.D. Dissertation, Temple Univ., Philadelphia, PA, May 1988, pp. 1–208 (also available through University Microfilms International, Ann Arbor, MI).

[4] H. Freeman and L. Garder, "Apictorial jigsaw puzzles: The computer solution to a problem in pattern recognition," *IEEE Trans. Electron. Comput.*, vol. EC-13, pp. 118–127, Apr. 1964.

[5] K. Hirota and Y. Ohto, "Image recognition in jigsaw puzzle assembly robot system (JPARS)," in Bull. Coll. Eng., Hosei Univ., Japan, pp. 87–93, Mar. 1986.

[6] K. Nagura, K, Sato, H. Maekawa, T. Morita, and K. Fujii, "Partial contour processing using curvature function—Assembly of jigsaw puzzle and recognition of moving figures," *Syst. Computing*,vol. 2, pp. 30–39, Feb. 1986.

[7] H. Blum, "A transformation for extracting new descriptors of shape," in *Proc. Symp. Models for Perception of Speech and Visual Form*,Weiant Whaten-Dunn, Eds. Cambridge, MA: MIT Press, 1967, pp. 362–380.

[8] E. Persoon and K. S. Fu, "Shape discrimination using Fourier descriptors," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 170–179, May 1977.

[9] M. K. Hu, "Visual pattern recognition by moment invariants," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 179–187, Feb., 1962.

[10] E. Wong and E. L. Hall, "Scene matching with invariant moments," *Computer Graphics, and Image Processing*, vol. 8, pp. 16–24, 1978.

[11] T. Nguyen and J. Sklansky, "A fast skeleton finder for coronary arteries," in *Proc. IEEE Int. Conf. Pattern Recog.*, Paris, France, Oct. 27–31, 1986, pp. 481–48.

[12] H. Blum, "A geometry for biology," in *Mathematical Analysis of Fundamental Biological Phenomena, Ann. New York Acad. Sci.*, vol. 231, pp. 19–30, 1974.

[13] You, Z. and K. Anil, "Performance evaluation of shape matching via chord length distribution," *Computer Vision, Graphics, and Image Processing*, vol. 28, pp. 185–198, Nov. 1984.

[14] J. L. Pfaltz and A. Rosenfeld, "Computer representation of planar regions by their skeletons," *Commun. ACM*, vol. 10, no.2, pp. 119–122, Feb. 1967.

[15] C. Arcelli, L. Cordella, and S. Levaldi, "From local maxima to connected skeletons," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-3, no. 2, pp. 134–143, Mar. 1981.

[16] U. Montanari, "A method for obtaining skeletons using a quasi-euclidean distance," *J. ACM*, vol. 16, pp. 534–549, 1969.

[17] B. Shapiro, J. Pisa, and J. Sklansky, "Skeleton generation from x,y boundary sequences," *Computer Vision, Graphics, and Image Processing*, vol. 15, pp. 136–153, 1981.

[18] L. Dorst, "Pseudo Euclidean skeletons," in *Proc. IEEE Int.Conf. Pattern Recog.*, Paris, France, Oct. 27–31, 1986, pp. 286–288.

[19] C. Lee, "Modified distance transform and linking algorithm for image skeletonization," in *Proc. SPIE*, vol. 415, Apr. 1983, pp. 147–154.

[20] B. Zvolanek, and C. Lee, "Image skeletonization for object position measurement," in *Proc. SPIE*, vol. 359, Aug. 1982, pp. 24–27.

[21] D. Lee, "Medial axis of a planar shape," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-4, no. 4, pp. 363–369, July, 1982.

[22] S. Peleg and A. Rosenfeld, "A min-max medial axis transformation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-3, no. 2, pp. 208–210, Mar., 1981.

[23] P. V. DeSouza, and P. Houghton, "Computer location of medial axes," *Comput. Biomed. Res.*, vol. 10, no. 4, pp. 333–343, Aug. 1977.

[24] G. Borgefors, "A new distance transformation approximating the euclidean distance," *Proc. IEEE Int. Conf. Pattern Recog.*, Paris, France, Oct. 27–31, 1986, pp. 336–338.

[25] P. Danielsson, "Euclidean distance mapping," *Computer Vision, Graphics, and Image Processing*, vol. 14, pp. 227–248, 1980.

An Algorithm for Point Clustering and Grid Generation

Marsha Berger and Isidore Rigoutsos

Abstract— **The paper describes a special purpose point clustering algorithm, and its application to automatic grid generation, a technique used to solve partial differential equations. Extensions of techniques common in computer vision and pattern recognition literature are used to partition points into a set of enclosing rectangles. Examples from two-dimensional (2-D) calculations are shown, but the algorithm generalizes readily to three dimensions.**

I. INTRODUCTION

This paper presents a special purpose clustering algorithm for the automatic generation of grids when solving partial differential equations using adaptive mesh refinement. Adaptive mesh refinement

Manuscript received April 25, 1990; revised January 26, 1991. This work was supported in part by the Air Force Office of Scientific Research under Contract No. AFOSR-86-0148, in part by the NSF Presidential Young Investigator Award ASC-8858101, in part by the Department of Energy Contract No. DEAC0276ER03077-V, and in part by Cray Research and Grumman Aerospace, who gave matching funds in support of the PYI award.

The authors are with the Courant Institute of Mathematical Sciences, 251 Mercer St., New York, NY 10012.

IEEE Log Number 9100401.

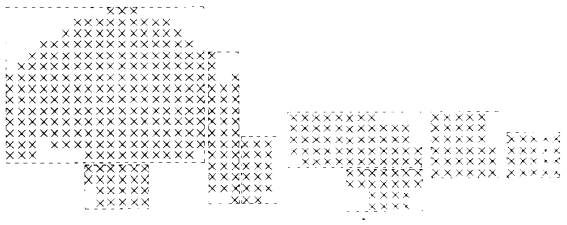

Fig. 1. A coarse grid with rectangular subgrids around the flagged points is shown. The coarse grid points with high error are marked with an "×".

methods can be summarized as follows. We are solving a partial differential equation using finite difference equations on a uniform rectangular grid. Often, the solution we are computing has sharp localized features that cannot be resolved using a uniform grid of practical size. The first step of adaptive mesh refinement is a mathematical "error estimation" procedure that "flags" grid points that have insufficient resolution [13]. The second step, which is the subject of this paper, is a "grid generation" procedure that places small finer grid patches over the coarse grid that cover the flagged points (see Fig. 1).

In our approach the flagged points are fist separated into clusters; each cluster is then covered with a single rectangle. Thus, the problem of automatic generation of fine grids is reduced to the problem of intelligent clustering of sets of points. This application is best illustrated by an example. Fig. 1 shows a coarse grid, along with the flagged points where the error estimate is too large. The rectangles produced by the algorithm are indicated in dashes. Each rectangle will become a finer grid with smaller mesh spacing.

The clustering algorithm must separate the points into distinct rectangles such that neighboring points are in the same rectangle (as much as possible), and all points are contained in some rectangle. We make no assumptions about the shape of the regions needing refinement; for example, oblique shocks, reflecting shocks, etc., lead to regions shaped like slanted lines or the letter "V". Our algorithm is completely general, and so there is no attempt to do a structural decomposition of shapes.

The grid generation problem then is to define an optimal set of rectangles enclosing all the flagged points. Certain factors make an enormous difference in the performance of these subgrids in solving the differential equations:

There should be as little unnecessarily refined area as possible.

Since the cost of the numerical integration procedure is proportional to the area of the rectangle, the smaller the better. Fig. 1 gives an example of a set of flagged points for which a few patches lead to much less refined area than if the whole grid were refined. This gain in efficiency is the purpose of adaptive methods. Some unnecessarily refined area (or inclusion of nonflagged coarse grid points in a new rectangle) is inevitable, since we are restricted to using rectangles. In addition, for numerical reasons the rectangles are oriented with the base grid rectangle. This is true even if the flagged points lie on a diagonal of the coarse grid, and could be perfectly enclosed by a rotated rectangle. (However, an algorithm that uses rotated rectangles is considered in [3]). Along these lines, if several rectangles are used to enclose the flagged points, their overlap should be minimal, to further reduce the computational time.

There should be as few rectangles as possible.

At the other extreme, we could put one tiny rectangle around each flagged point. Many of these tiny rectangles would share a common boundary segment. However, there is boundary overhead associated with each rectangular subgrid that should also be minimized, along with the area. In addition, these procedures will be used on vector processors, which favor larger vector lengths and therefore larger rectangles. (We could worry further about this, for example by trying to maximize the length in a particular coordinate direction, but we will not consider such machine specific details here). These first two criteria are often at odds with each other, however our final algorithm strikes a nice balance between them.

The rectangles should "fit" the data.

This is hard to make absolutely precise, but for example, if a person were to put the rectangles around the points by hand, using whatever clustering or partitioning the brain uses, it would "look right." Although this is not essential for accurate numerical integration on the rectangles, we prefer the adaptively generated rectangles to be pleasing. Finally,

The algorithm should be fast.

This algorithm is repeated every few timesteps, or hundreds of times during any particular numerical simulation, and should therefore be fast relative to the time needed to take a integrate the solution on the resulting grids.

Our solution to this rectangle-fitting problem employs ideas from the pattern recognition and computer vision literature. A combination of one dimensional signatures and zero crossings of second derivatives is used to partition the flagged points into disjoint rectangles. Although reminiscent of structural pattern recognition approaches, it should be stressed that our algorithm does not attempt to perform structural decomposition of its input, and our decompositions do not necessarily generate meaningful parts. Our examples are all in two dimensions, but the algorithm generalizes readily and has proven effective in three dimensions too. Before describing the algorithm, we give a little background and discuss some other approaches we tried and discarded.

Previous Algorithms

Our previous algorithms for this problem can be summarized as being of two main types: bottom–up or top–down. The top–down approach is based on a bisection method. It can be viewed as a form of divisive hierarchical clustering [6]. Initially, the flagged points of the grid are surrounded by a single rectangle, and its *efficiency* is computed. Here, we define the efficiency of a rectangle as the ratio of flagged points to the total number of coarse grid points in the new rectangle. This is one of the key parameters behind our algorithm, and it is easily computed. If the efficiency is above a preselected threshold, the rectangle is accepted and the algorithm stops. Otherwise, we bisect the longest direction of the rectangle, and generate two smaller rectangles. This process is repeated recursively on each of the two new rectangles. When the algorithm terminates, all of the rectangles are guaranteed to have acceptable efficiency ratings. However, hierarchical clustering methods are known to create clusters even if no natural clusters exist [1], [7], [9]. In addition, since the bisection approach uses no information about the locations of the flagged points, a nonoptimal grid hierarchy is generally created. To alleviate this, we usually follow the bisection step with a merging step, where neighboring rectangles are merged into larger ones if the result continues to be acceptably efficient. This merging step is what leads to the problem of overlapping rectangles.

The bottom–up approach starts at the grid point level. The flagged points are organized into a minimal spanning tree, so that each point is connected to its nearest neighbor. Neighboring branches of the tree are merged, either one point at a time, or a front at a time, as long as

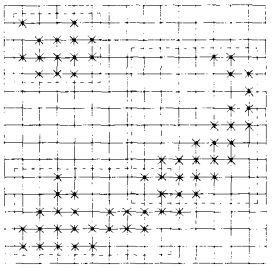

Fig. 2. Nearest neighbor clustering is sufficient in simple cases, e.g., the top left cluster.

the resulting grid is efficient. Although philosophically appealing, this algorithm actually performs much worse than the top–down bisection algorithm. A fundamental problem is the nonuniqueness of the minimal spanning tree. Also, the algorithm suffered from the hill-climbing problem of getting stuck in local minima; it was very sensitive in the first few steps to the initial direction of growth of the clusters, and tended to stop prematurely, although larger and acceptably efficient grids were just several branches away. In that case, it had to be followed by a merging procedure as well.

In practice, both approaches were preceded by a "nearest neighbor" clustering algorithm. The purpose of this was to separate flagged points when possible into isolated islands (see Fig. 2). This sometimes produces acceptable clusters by itself, but fails to help when the flagged points formed elongated, curved shapes. Thus, it was followed by either the bisection or minimal spanning tree algorithm. Summarizing, both of these algorithms produced less than optimally efficient grids that overlapped too much. Better grids were easily created by hand.

II. Toward an Efficient Algorithm

In a more general form, the grid generation algorithm should cluster a set of m flagged points into k clusters, where k is either specified a priori or is determined by the algorithm itself. This special type of clustering is called partitioning [1]. In the experiments that follow, below each example we show the total number of coarse grid points in the generated sub-rectangles. The ratio of the total number of flagged points to the total number of refined points (counting overlapping areas twice) provides a simple measure of the effectiveness of our partitioning algorithms. Although it doesn't include all of our evaluation criteria, it does provide a good measure of the cost of doing the computation on the new fine grids, and adds a more objective means of evaluating the performance of the partitioning algorithms.

A First Approach: Local Maxima in Two-Dimensional (2-D) Grids

Our first approach considered the question of how to choose a set of k "seed points" around which k clusters would be built. Initially, each of the grid points is given a value: "1" for the flagged, and "0" for the nonflagged ones. These grid values, viewed as a binary image, $I(x, y)$, are then preprocessed by convolving it with either an "averaging" or a "low-pass" filtering template, (see Fig. 3) [2], [8], [11]. This operation results in a nonconvex function, $\hat{I}(x, y)$, whose local maxima, determined by the Sobel operators [2], [11] of Fig. 4, compose the set of "seed points" around which we build the clusters.

Three partitioning algorithms were tested using the seeds found above: the standard k-means algorithm, its converging variant, and a k-means variant where no updating of the centroids takes place [1], [7]. These algorithms are outlined in the Appendix.

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

(a)

1	1	2	1	1
1	2	4	2	1
2	4	8	4	2
1	2	4	2	1
1	1	2	1	1

(b)

Fig. 3. The two filtering templates. (a) Averaging. (b) Low-Pass.

-1	0	1
-2	0	2
-1	0	1

(a)

1	2	1
0	0	0
-1	-2	-1

(b)

Fig. 4. The Sobel gradient operators. (a) $\partial/\partial x$. (b) $\partial/\partial y$.

Figs. 5 and 6 show graphically the output of the three algorithms on two sample data sets. The three algorithms exhibit the same overall behavior. The seeds are sensibly chosen, but in all the test cases the resulting rectangles overlap excessively. The next method tries to reduce the number of seeds, keeping only the best, and thus hopefully reducing the overlapping.

A Second Approach: Local Maxima in Signature Arrays

Signatures have been known in computer vision and pattern recognition for many years [2], [8], [11]. Able to encapsulate gross characteristics of a shape, and computationally simple, signatures proved very useful for establishing preliminary landmarks in images; these landmarks in turn led to a subsequent reduction of the search effort. Given a continuous function, the horizontal and vertical signatures, $H(x)$ and $V(y)$ are defined as

$$H(x) = \int_y f(x, y)\ dy$$

and

$$V(y) = \int_x f(x, y)\ dx$$

respectively.

First, the horizontal and vertical signatures of the image are computed. The resulting one-dimensional (1-D) arrays are low-pass filtered [8], and subsequently searched for local maxima. Let M and N be the two sets of maxima. As can be seen from Fig. 7, some of the tuples of the Cartesian product $M \times N$ do not correspond to flagged point regions. After discarding all such tuples, we are left with precisely the coordinates of the starting seeds.

With this choice of seeds, we again employed the three partitioning techniques (k-means, converging k-means, and the no updating variant). Figs. 8 and 9 show the results. This algorithm considerably reduced overlapping, making this approach superior to using the local maxima of $\hat{I}(x, y)$ for the seed points. Unfortunately, the generated subgrids were still not the most efficient ones; better choices were clearly possible. Some observations based on extensive experimentation were made: 1) the nonconverging variants outperformed the

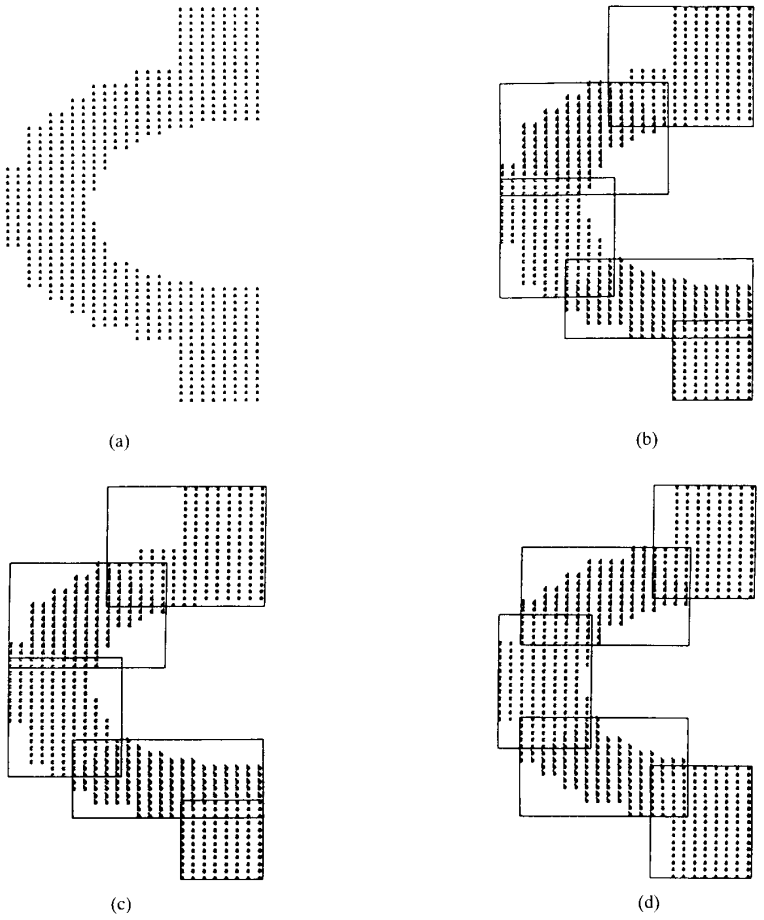

Fig. 5. The seeds are local maxima of a 2-D function. (a) Input, # flagged pts=974. (b) k-means, area of rectangles=1624, ratio=60.0%. (continued). (c) Converging k-means, area of rectangles=1580, ratio=61.6%. (d) No centroid updating, area of rectangles=1687, ratio=57.7%.

converging k-means, 2) overlapping was minimal when no updating of the centroids occurred, and 3) the distance metric (Manhattan Block versus Euclidean) had no appreciable effect on the results.

Apparently, the use of local maxima in signatures did not capture enough of the underlying structure. In the next section, we use signatures in a different way to partition the flagged points into clusters.

III. The Algorithm

Our best algorithm uses ideas related to edge detection. One of the many approaches to the edge detection problem is the one suggested by [12]. Based on the psychophysical and neurophysiological experiments of [5], the method consists of first convolving the original image against a Gaussian kernel and then computing the Laplacian of the result; the intensity discontinuities are associated with those positions where the Laplacian is equal to 0 (zero crossings).

In what follows, the input grids are viewed as binary images in the sense of Section II-A. The edges will now be located at those positions of the grid where a transition from a flagged point region to a nonflagged one occurs. The most prominent such transition represents a "natural" line with respect to which the original grid can be partitioned. For the example depicted in Fig. 10, the line (e) represents such a transition.

The problem is that of determining such transitions. Although the Marr-Hildreth method would be a potential candidate, it cannot easily be employed toward this end for two reasons. First, its output is a collection of Boolean values at the different grid locations: TRUE if a zero crossing exists at that location, FALSE otherwise; these Boolean values need to be combined in order to generate an actual edge, a not so trivial task. Second, since the Laplacian operator is isotropic, the "natural" line with respect to which the original grid could be split will not, in general, be parallel to the sides of the grid. (We require the sides of the rectangle to be parallel to the sides of the original grid).

Signatures again hold the answer: the idea is to look for zero crossings in the second derivative of a signature (inflection points). This idea borrows from both the signature approach and the Marr-Hildreth method, except that we do not convolve with a Gaussian filter.

In general, there is more than one inflection point in a given signature array. For our purposes however, we select one inflection point at a time, corresponding to the most prominent edge. Its location is determined by searching both the horizontal and vertical signature arrays for the inflection point with the largest local change of values. This is indicated in Fig. 11, where the row (column) labeled Σ contains the horizontal (vertical) signature, and Δ indicates the Laplacian of the signature in the appropriate direction.

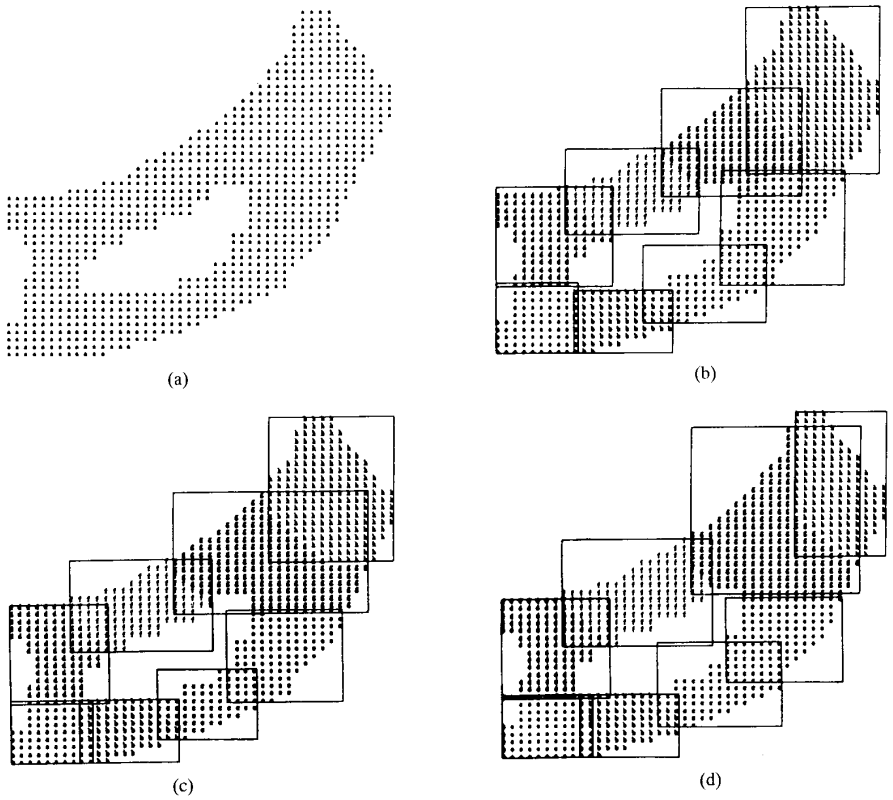

Fig. 6. The seeds are local maxima of a 2-D function. (a) input, # flagged pts = 704. (b) k-means area of rectangles =1034, ratio=68.1% (continued). (c) converging k-means, area of rectangles=1020, ratio=69.0%. (d) no centroid updating, area of rectangles=1000, ratio=70.4%.

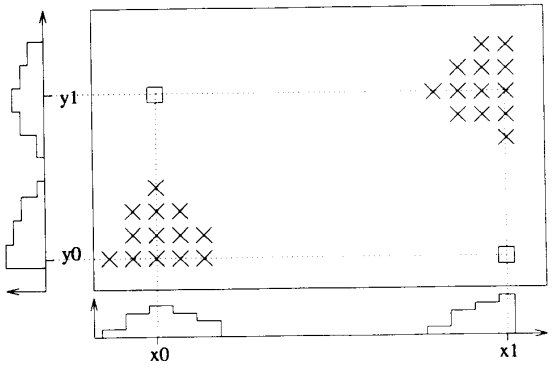

Fig. 7. Seed values are obtained from the maxima of the one dimensional signature arrays at x_0, x_1 and y_0, y_1. However, their Cartesian product does not always correspond to a region with flagged points.

The input grids are also expected to contain isolated regions of flagged points (islands), making it necessary that both signature arrays first be searched for chains of 0's, or "holes" (see Fig. 10). The occurrence of such holes provides obvious choices for splitting the input grid into a number of rectangles, and is exploited before any attempt at locating inflection points is made. If holes are found, we use them before computing any inflection points. We proceed by applying our algorithm only to those rectangles that are still inefficient.

We now give a high level description of the actual algorithm, followed by sample runs in Figs. 12 to 16 on a number of real and synthetic problem cases.

```
BEGIN
  i = 1;
  while ( i  number_of_rectangles ) do
    if (rectangle efficiency < threshold )
     then
      compute signatures ;
      find the best place to split R_i
       (either a hole or inflection pt.) ;
      if ( found a place to split ) then
        split rectangle in two ;
        append new rectangle to end
         of list of rectangles ;
      else
        i ← i + 1 ;
      endif
    else
      i ← i + 1 ; ( consider the next
       rectangle on the list )
    end if
  end while
END
```

IV. Comments on the Algorithm and Its Performance

While most of the time the algorithm performs exceptionally well, sometimes it generates anomalous and/or nonoptimal rectangles. As

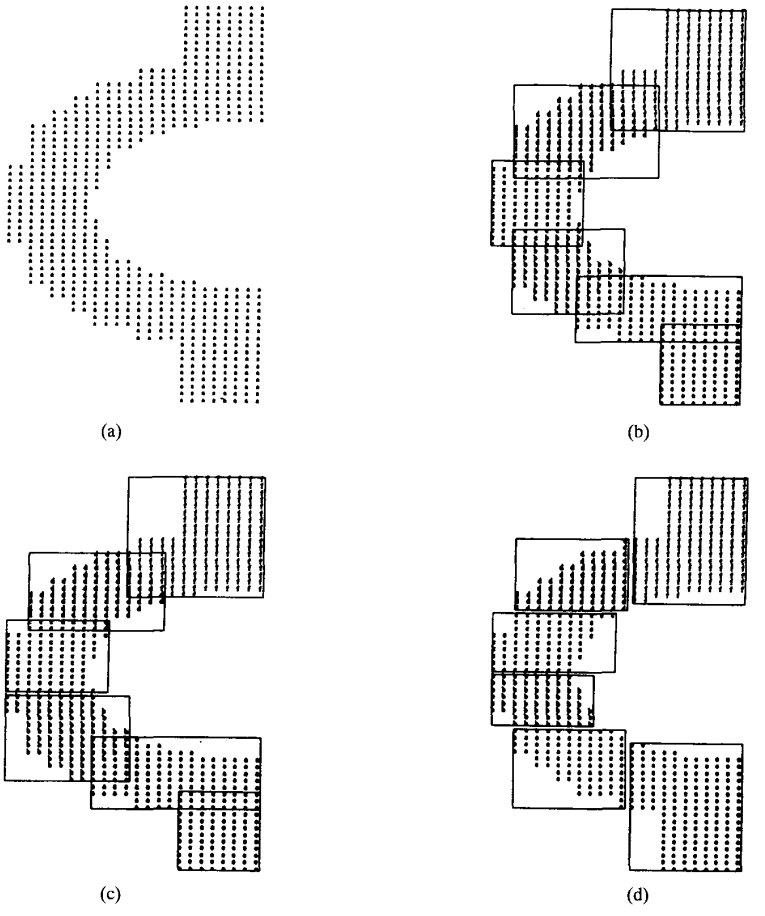

Fig. 8. The seeds are chosen from local maxima of signature arrays. (a) input, # flagged pts=704; (b) k-means, area of rectangles=946, ratio=74.4%. (continued) (c) converging k-means, area of rectangles=928, ratio=75.8%; (d) no centroid updating, area of rectangles=859, ratio=81.9%.

we will see, many of these can be eliminated by simple changes in the algorithm. The anomalies fall into several general categories, which we illustrate by picture. The basic algorithm description has two exit points: a rectangle is "accepted" either because its efficiency is already above threshold, or because it cannot be further split using either of the two methods (i.e. holes and inflection points). As a result, the efficiency of some of the generated subgrids may be below the preset threshold. This is the case with all grids associated with regions that form an angle with the horizontal (see Figs. 6 and 9). The inefficiency approaches its maximum as angles approach 45 degrees.

Another problematic grid is the one appearing in Fig. 17. Neither the horizontal nor the vertical signatures will contain any holes or inflection points, and this will be true for all nonzero values of a, b. In these cases the bounding rectangles will have an efficiency of precisely 50%. An ordinary bisection step could be easily incorporated here to increase the efficiency to 100%. A similar arrangement of flagged points which we have encountered in our experiments (see Fig. 18) leads to another kind of nonoptimal choice. One way around this is to used weighted second derivatives, scaling the Laplacian by the number of flagged points. This leads to a correct choice for the inflection point in Fig. 18.

A modification of our algorithm that covers both anomalous cases is to compute the sum of the absolute value of the gradient, and difference the results to get the second derivative. The most robust solution to this problem is still an open question.

A seemingly problematic case is shown in Fig. 19. Here it appears as if the algorithm made a nonoptimal decision by unnecessarily splitting rectangle $R1$ (dotted line) into two smaller subrectangles. However, careful inspection shows that rectangles $R1$ and $R2$ could not have been generated without introducing an overlapping region.

Tight bounds for the running time of the described algorithm are very hard to establish, since the precise flow of the algorithm is input dependent. However, it should be clear that the running time is $O(k(P + N + M))$, where k is the total number of grids upon termination of the algorithm, and P is the number of flagged points. The P term comes from computing the signatures of the points (by traversing the list of flagged points). The N (resp. M) term comes from the linear search that determines the best inflection point. The above bound is far from optimal, and the algorithm performs very well in practice. Preliminary three dimensional results also show great performance.

V. Conclusion

We have described a new and efficient algorithm for point clustering and adaptive grid generation. The algorithm's performance

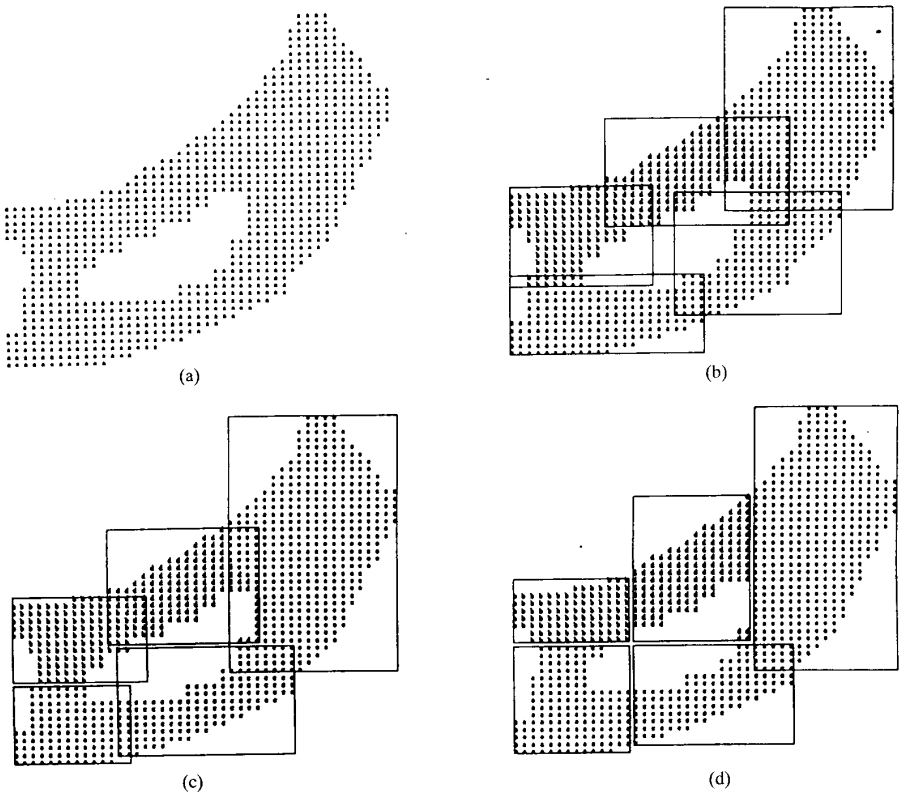

Fig. 9. The seeds are chosen from local maxima of signature arrays. (a) input, # flagged pts=974; (b) k-means, area of rectangles=1721, ratio=56.6%. (continued) (c) converging k-means, area of rectangles=1649, ratio=59.1%; (d) no centroid updating, area of rectangles=1494, ratio=65.2%.

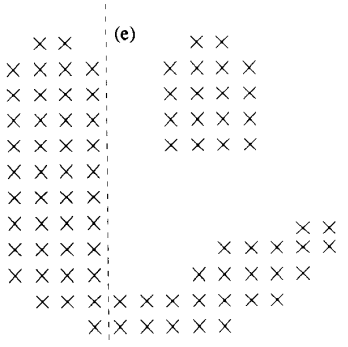

Fig. 10. The rectangle is partitioned at line (e); after this, an isolated cluster can be detected for further partitioning.

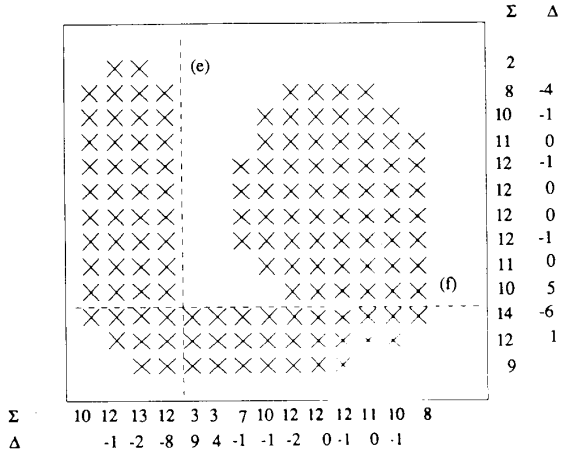

Fig. 11. The biggest inflection point occurs at line (e), and makes the most efficient rectangles.

has been demonstrated through a series of graphs showing results obtained with both synthetic and actual 2-D inputs. Preliminary experiments with three-dimensional (3-D) problems also show a considerably improved performance over the previous approaches. In general, the efficiency of the enclosing rectangles for our applications has been very high, typically ranging between 85% and 100%, with the exception of the problematic cases illustrated in Figs. 17–19. It is surprising how effectively the algorithm performs on multidimensional data, even though it is based on Cartesian coordinate directions. Finally, this algorithm may also prove useful in other applications with binary image data, for example in generating bounding rectangles for computer graphics applications. A rectangle fitting algorithm has also been used in conjunction with a pattern recognition system for understanding Japanese business cards [10].

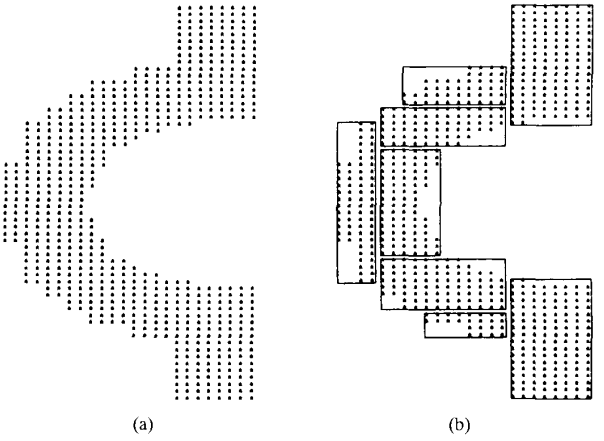

Fig. 12. An example of the final algorithm. (a) input, # flagged pts = 704. (b) area of rectangles=756, ratio=93.1%.

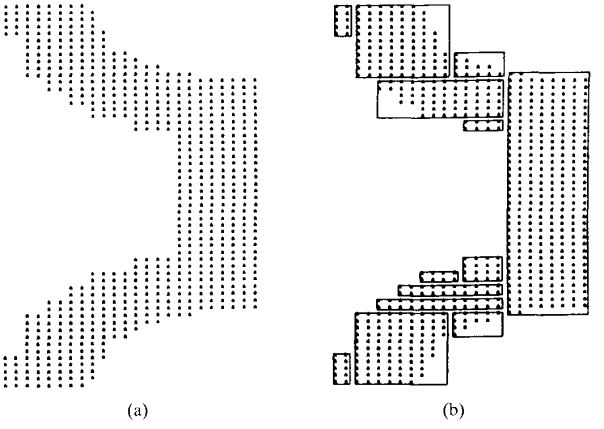

Fig. 14. An example of the final algorithm. (a) input, # flagged pts=686. (b) area of rectangles=694, ratio=98.8%.

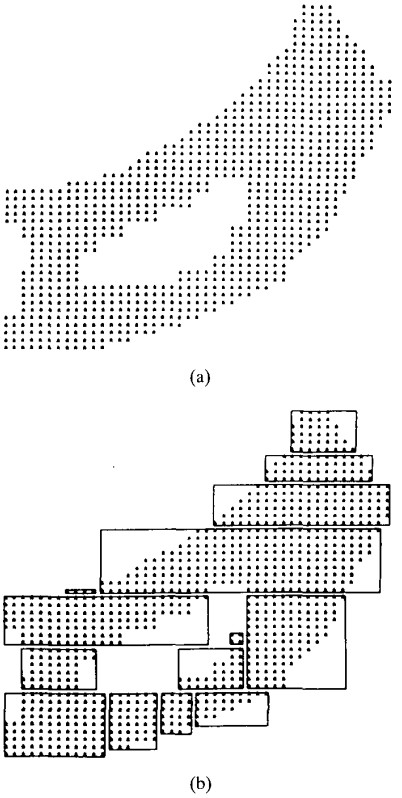

Fig. 13. An example of the final algorithm. (a) input, # flagged pts=974. (b) area of rectangles=1182, ratio=82.4%.

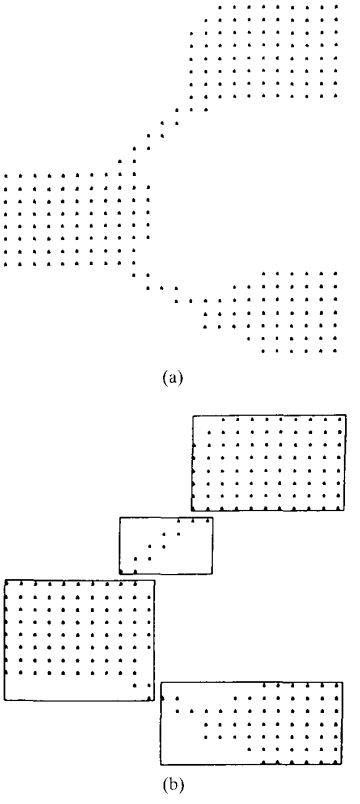

Fig. 15. An example of the final algorithm. (a) Input, # flagged pts=273. (b) Area of rectangles=324, ratio=84.3%.

APPENDIX

McQueen's k-means Partitioning Algorithm

1) Form k single member clusters each one containing precisely one of the k starting seeds. The clusters' centroids originally coincide with the starting seeds.
2) Assign each of the remaining data points to the cluster with the nearest (with respect to an appropriate distance metric) centroid recomputing the gaining cluster's centroid after each assignment.
3) Assume the cluster centroids are fixed this time, and reassign each of the data points to the cluster with the nearest centroid (one pass through the data).

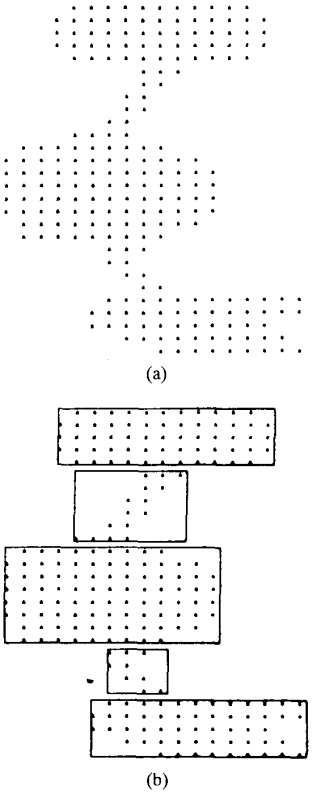

Fig. 16. An example of the final algorithm. (a) input, # flagged pts=261; (b) area of rectangles=292, ratio=89.4%.

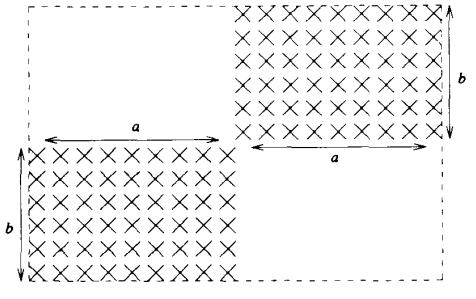

Fig. 17. There are no inflection points in the signature arrays.

McQueen's Converging k-means Partitioning Algorithm

1) Form k single member clusters each one containing precisely one of the k starting seeds. The clusters' centroids originally coincide with the starting seeds.
2) Assign each of the remaining data points to the cluster with the nearest (with respect to an appropriate distance metric) centroid recomputing the gaining cluster's centroid after each assignment.
3) For each data point compute its distance to all the cluster centroids; if the nearest centroid corresponds to a cluster other than the point's actual parent cluster reassign the point; recompute the centroids of both the gaining and losing clusters.
4) Repeat step 3) until a full sweep through the data does not induce further changes in the points' memberships.

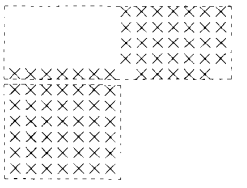

Fig. 18. This set of points leads to a nonoptimal partition, unless the second derivatives are scaled.

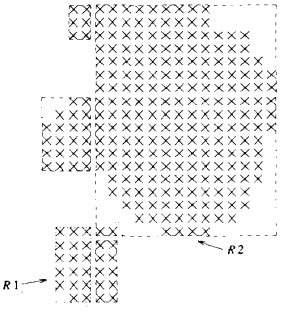

Fig. 19. An unexpected set of rectangles.

Non-Updating Variant of k-means Partitioning Algorithm

1) Form k single member clusters each one containing precisely one of the k starting seeds. The clusters' centroids remain fixed throughout the algorithm.
2) Assign each of the remaining data points to the cluster with the nearest (with respect to an appropriate distance metric) centroid but do not update the gaining cluster's centroid (one pass though the data).

REFERENCES

[1] M. Anderberg, *Cluster Analysis for Applications.* New York: Academic, 1973.
[2] D. Ballard and C. Brown, *Computer Vision.* Englewood Cliffs, NJ: Prentice Hall, 1982.
[3] M. Berger, "Data structures for adaptive grid generation," *SIAM J. Sci. Stat. Comp.*, vol. 7, July 1986.
[4] M. Berger and P. Colella, "Local adaptive mesh refinement for shock hydrodynamics," *J. Comp. Phys.*, vol. 82, May 1989.
[5] F.W.C. Campbell and J. Robson, Application of Fourier analysis to the visibility of gratings," *J. Phys.*, vol. 197, 1968.
[6] R. Duda and P. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.
[7] J. Hartigan, *Clustering Algorithms.* New York: Wiley, 1975.
[8] B. Horn. *Robot Vision.* Cambridge, MA: MIT Press, 1986.
[9] A. Jain and R. Dubes, *Algorithms for Clustering Data.* Englewood Cliffs, NJ: Prentice Hall, 1988.
[10] K. Kise, K. Yamada, N. Tanaka, N. Babaguchi, and Y. Tezuka, "Visiting card understanding system," *Proc. Int. Conf. Pattern Recog.*, 1988.
[11] M. Levine, *Vision in Man and Machine.* New York: McGraw-Hill, 1987.
[12] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. Royal Soc. London* vol. 207, 1980.
[13] J. Oliger, "Approximate methods for atmospheric and oceanographic circulation problems," *Lecture Notes in Physics 91*, Glowinski and Lions, Eds. New York: Springer-Verlag, 1979.