by the fact that although conservative estimates slow down the convergence, eventually they would lead to correct classifications. Overestimation, however, might lead to a premature erroneous classification. It appears, therefore, that decision makers would be well advised to use conservative estimates in the event of doubtful situations.

The main implication is that Bayesian systems which are based on limited data or subjective probabilities are expected to have a high percentage of correct classification despite the fact that the prior and conditional probabilities they use may deviate rather significantly from the true values.

REFERENCES

[1] M. Ben-Bassat, R. W. Carlson, V. K. Puri, M. D. Davenport, J. A. Schriever, M. Latif, R. Smith, L. D. Portigal, E. Lipnick, and M. H. Weil, "Pattern-based interactive diagnosis of multiple disorders: The MEDAS system," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 148–160, Mar. 1980.

[2] R. M. Hogarth, "Cognitive procedures and the assessment of subjective probability distributions," *J. Amer. Statist. Ass.*, vol. 70, pp. 271–294, 1975.

[3] S. Lichtenstein, B. Fischhoff, and L. D. Phillips, "Calibration of probabilities: The state of the art," in *Decision Making and Change in Human Affairs*, H. Jungerinan and G. deZeeuw, Eds. Amsterdam, The Netherlands: Reidel, 1978.

[4] R. S. Lindley and L. B. Lusted, "Reasoning foundations of medical diagnosis," *Science*, vol. 130, pp. 9-21, 1959.

[5] P. Slovic, B. Fischoff, and S. Lichtenstein, "Behavioral decision theory," *Annu. Rev. Psychol.*, vol. 28, pp. 1-39, 1977.

Finding the Extrema of a Region

WESLEY E. SNYDER AND D. ALLEN TANG

Abstract—**The computational difficulty of finding the extreme points (those two points furthest from each other) in a region of an image is examined. Various approaches are described, including computation of convex hulls, sophisticated search strategies, and a statistical method. Suitability of the various methods to particular applications is discussed.**

Index Terms—**Computer vision, extrema, pattern recognition, region growing, scene analysis.**

INTRODUCTION

Consider, for a moment, the following scenario: "I" (who happen to be simulating a digital computer) am looking for a region in an image. I am only willing to accept regions which have a particular appearance—namely, such regions should be long, thin, and oriented in a particular direction. How can I quantify such vague terms as "long," "thin," and what precisely is the orientation of a region? After all, a region is simply a list of points which was output from my region growing [1], [7] routine. There are, of course, a large number of methods described in the literature for describing shape of regions (see [9], [10]). One approach to solving such problems makes use of the location of the "extremes" or diameter of the region; that is, those two points, A and B, whose mutual distance $d(A, B)$ is greater than or equal to the distance between any other two points in the region.

Manuscript received March 19, 1979; revised September 3, 1979.

The authors are with the Image Analysis Group, Department of Electrical Engineering, North Carolina State University, Raleigh, NC 27607.

This calculation can find applications when it is necessary to quantify the "shape" of a region in an image, as might occur in classification of parts on an assembly line [3], optical character recognition [5], etc. We encounter it in an application [2] which required that the orientation of region be determined.

We assumed we have a region in a two-dimensional image plane described by a set of points R. The problem then is to find two points, $A, B \in R$, such that $\forall\ p_1, p_2 \in R\,[d(A, B) > d(p_1, p_2)]$. If the set R is small (say, 10–20 points) a simple comparison of every point with every other point is the most straightforward approach. However, when R becomes larger, the number of comparisons needed grows as $n(n-1)/2$ or $0(n^2)$, and rapidly becomes prohibitive.

In this paper, we survey four techniques for solving this problem. The first uses eigenvalue analysis to find the "best" (in a generalized least-squares sense) estimate of the major axis of the region, and defines the extrema as being the two points furthest out in the direction of that axis. The second method uses hill climbing and avoids the problem of local maxima by first ensuring that the region is convex. The third and fourth approaches are heuristic methods which are, (in the case of Method 3) very fast but may become trapped in local maxima or somewhat slower and much more complex, but guaranteed to converge (Method 4).

I. THE PRINCIPAL-AXIS APPROACH

In this method we first find the major axis of the region and those two points on the boundary of the region which are closest to that axis. In this approach, minor deviations, such as the spur shown in Fig. 1, will be ignored, even though they may actually contain one of the extreme points.

The first step in the process is calculation of the major axis. This is performed by a minimization-of-squared-error technique. It is important that the minimization of error be independent of the coordinant axes; therefore, we use the eigenvector line fitting technique rather than the conventional MSE technique. (This is well described in [4].) We define a line to be the best representation of the major axis if it minimizes the sum of squares of the perpendicular distances from the points in the region to the line.

Let us assume the region R is described by a set of n points $R = \{(x_i, y_i) | i = 1, \cdots, n\}$. Let the point (x_i, y_i) be denoted by the vector v_i, and let d_i be the perpendicular distance from v_i to the major axis. Thus, the major axis of the region is the line which minimizes

$$d^2 = \sum_{i=1}^{n} d_i^2.$$

It is easily shown that the major axis must pass through the center of gravity of the region; thus it is necessary only to find the slope of the axis. Since the line passes through the center of gravity, let us take that point as the origin of our coordinate system. Then the problem becomes: given n points with zero mean, find the line through the origin which minimizes d^2.

This turns out to be an eigenvector problem. Specifically, the matrix $S \triangleq \sum_{i=1}^{n} v_i v_i^t$ is the scatter matrix of the region, and the eigenvector corresponding to the largest eigenvalue of S will be parallel to the principle axis of the region. Thus one can find the major axis by:

1) relocating the origin by subtracting the center of gravity from each point;

2) finding the principle eigenvector of the scatter matrix of this modified set of points; and

3) solving for the major axis as the line through the center of gravity parallel to the principle eigenvector.

0162-8828/80/0500-0266$00.75

Fig. 1. A region with a spur.

Having found the major axis, one then treats each point on the boundary as a vector and projects each onto the major axis. The extrema are then those two points on opposite sides of the center of gravity whose projections onto the major axis are maximum in length. (The extrema are not necessarily unique.) This approach yields a solution which is an accurate representation of the "shape" of the region in the least-mean-squared-error sense. It may or may not actually find the two points whose mutual distance is maximum. In many applications, an approximation like this is exactly what is needed. However, occasionally one encounters regions (Fig. 1) with spurs on them, where the spur may be relevant. In this case, it is necessary to use an algorithm which will find the actual extrema.

II. The Convex Hull

If we are fortunate enough to have a convex region to analyze, and we know the boundary of the region, the following iterative hill-climbing strategy will work.

1) Choose a point on the boundary, call it P_0.

2) Perform a linear search of the boundary points, testing adjacent points in turn, searching for the point maximum distance from P_0. Call the new point P_1. For P_1, $\forall x \in R [d(P_0, P_1) \geq d(P_0, x)]$.

3) Starting at P_0, search left and right for the point at the greatest distance from P_1. We need search only in the direction in which distance is increasing. If distance decreases in both directions, stop. P_0 and P_1 are the extrema; also go to step 4).

4) We find a new point P_2, such that $d(P_1, P_2) > d(P_1, P_0)$, then assign

$P_0 \leftarrow P_1$

$P_1 \leftarrow P_2$ go to C.

Since this method goes around the boundary only in the direction of increasing distance, it avoids comparing the same pair of points twice and hence operates in $0(n)$ time.

This method also requires that boundary points be known. This is often the case if the list of points in the region is derived by region growing. In any case, the boundary can always be found.

An alternative method is given by Shamos [11] which requires more calculations, but (in general) fewer iterations.

This algorithm converges rapidly (if the region is convex) to the extreme points. Unfortunately, for regions with concave areas along the boundary, it can get caught in a local maximum. There are several ways around this problem. First, the algorithm in Section III is less likely to get caught in a local maximum than this simple search, and the algorithm in Section IV is guaranteed to converge. One could, as an alternative, compute the convex hull and apply this procedure to it.

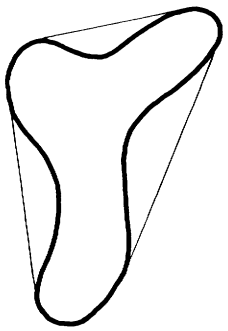

Fig. 2. A region and its convex hull.

The convex hull is best described as the figure which would result if one stretched a rubber band around the figure (Fig. 2). Since digital figures are not continuous, some care must be exercised in the computation of this figure [5], [7], [8].

Given the convex hull, one can find the extreme points by using the hill-climbing algorithm given above. The computational burden of the convex hull together with the hill-climbing can be quite high ($0(n \log n)$ [11]); however, having the convex hull can be desirable as one may want to use it later.

III. Search Strategies

We have been using the following algorithm for several years [2] quite successfully.

1) Choose a point on the boundary, call it P_0.

2) Find the point on the boundary at a maximum distance from P_0. Call it P_1.

3) Compute the midpoint, M, of P_0, P_1.

4) Find the point P_2 on the boundary, at a maximum distance from M.

5) Find the boundary point P_3, a maximum distance from P_2.

6) If $d(P_2, P_3) \leq d(P_0, P_1)$, stop. P_0 and P_1 are the extrema. If $d(P_2, P_3) > d(P_0, P_1)$, assign $P_0 \leftarrow P_2, P_1 \leftarrow P_3$, go to 3).

This technique is easy to program and converges rapidly. We have had good luck with it. Unfortunately, it is not guaranteed to converge to the global extrema, although there is a high likelihood that it will do so.

IV. A Global Strategy

An extension of the previous algorithm provides a strategy which is guaranteed to converge. In addition, this algorithm provides a mechanism for rapidly narrowing the search space.

First, define a linear search function $M(P_i, R)$ which returns the point $P_{i+1} = M(P_i, R)$ such that $\forall x \in R [d(P_i, x) \leq d(P_{i+1}, P_i)]$.

Choose an arbitrary point $P_1 \in R$, find $P_2 = M(P_1, R - \{P_1\})$. Next find $P_3 = M(P_2, R - \{P_1, P_2\})$. There are three cases shown below (Figs. 3–5).

Case 1:

$$d(P_1, P_2) > d(P_2, P_3).$$

Case 2:

$$d(P_1, P_2) = d(P_2, P_3).$$

Case 3:

$$d(P_1, P_2) < d(P_2, P_3).$$

In Case 3, compute a new P_3 called P_3' so that $d(P_2, P_3') =$

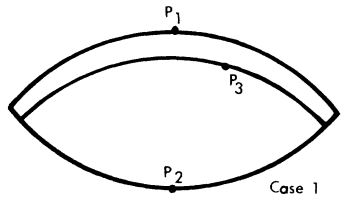

Fig. 3.

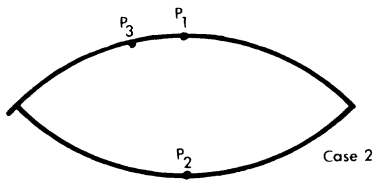

Fig. 4.

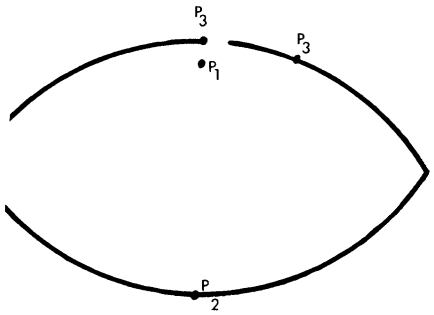

Fig. 5.

$d(P_2, P_3)$ and P_1, P_2, and P_3' are collinear. We, therefore, have a symmetrical lens shape which encloses all the points in R using at most two linear searches.

Our heuristic states that the "best" direction in which to search is perpendicular to P_1, P_2 (or P_2, P_3' in Case 3).

Compute the apexes α_1 and α_2 as shown in Fig. 6. Then find $B_1 = M(\alpha_1, \{R - \{P_1, P_2\}\})$. If $d(\alpha_1, B_1) \leqslant d(P_1, P_2)$, stop, else partition $R - P_1, P_2$ into two mutually exclusive regions R_1 and R_2. $R_1 = \{x \in R - \{P_1, P_2\} | d(x, P_1) > d(P_1, P_2)\}$. Find $B_2 = M(\alpha_2, R - R_1)$, and find: $R_2 = \{x \in R - \{P_1, P_2\} - R_1 | d(x, \alpha_2) > d(P_1, P_2)\}$.

Note: 1) In general, $R_1 \cup R_2 \subset R$. However, $R - \{R_1 \cup R_2\}$ contains no points of interest.

2) If either $R_1 = \phi$ or $R_2 = \phi$, we can stop and P_1, P_2 is the diameter.

If both R_1 and R_2 are nonempty, we can define them as "antipodal regions" in the following sense: if a diameter exists greater than $d(P_1, P_2)$, one endpoint must lie in R_1 and the other in R_2, which effectively cuts the search space in half.

Two new points, β_{11} and β_{12}, are computed by swinging an arc with center at α_1 and radius $d(\alpha_1, B_1)$ to intersect the lens. Similarly, β_{21} and β_{22} are computed.

Note that $d(\beta_{21}, \beta_{12}) = d(\beta_{11}, \beta_{22})$ and this is an upper bound on the diameter. In a digital picture, having an upper limit on the diameter may allow an earlier stopping of the algorithm, since if we know a diameter candidate, and we know the upper bound, if these two values differ by less than 1.414 (the diagonal of a pixel), there is no need to search further.

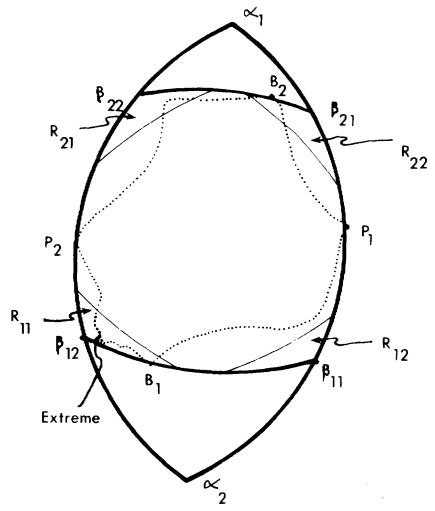

Fig. 6. B_2 is one of the extrema. The other extrema is an element of R_{11}, as shown.

Compute $r = \max(d(B_1, B_2), d(P_1, P_2))$. Use this as a radius with which to draw arcs. Using β_{21} as center and r as radius, partition R_1 into two regions, R_{11} and $R_1 - R_{11}$. Similarly, use β_{22} as center and find $R_{12} \subset R_1$. Note that $R_1 - \{R_{11} \cup R_{12}\}$ contains no points of interest. Similarly, find R_{21} and R_{22} as shown. If $R_{21} = \phi$ and $R_{11} \cap R_{12} = \phi$, then R_{21} and R_{12} is an antipodal pair as is R_{11} and R_{22}. In any case, $R_{21} \cup R_{22}$ is antipodal with $R_{11} \cup R_{12}$. These antipodal regions will be our search space in the next phase.

The strategy to this point has either identified the extrema or it has provided other useful results, specifically:

1) an upper bound on the diameter has been derived,

2) all points within a very large area have been eliminated as candidates for extrema,

3) the remaining points have been partitioned into antipodal regions.

At this point, if less than K points remain (which is most often the case), the most appropriate technique is to compute convex hulls. (The optimal choice of K is a function of region topology. It has been our experience that $K = 50$ seems to work well.) This computation is aided by the observations that:

1) if either of an antipodal pair of regions is empty, the other may be eliminated;

2) the extrema must lie on opposite sides of the convex hull of the antipodal regions, eliminating a great many more points, since we are able to compute convex hulls of smaller regions.

If, on the other hand, many more points remain, the algorithm can be invoked recursively using the antipodal pairs of regions as subject areas, and choosing new starting points as those points closest to β_{21} and β_{12}, or β_{11} and β_{22}.

A. Computational Complexity

For R with N points, the blind exhaustive search takes C_2^N distance calculations and comparisons. Our search algorithm is exhaustive (hence guarantees convergence), but intelligent by taking the global shape of the region R into consideration. The initial search space R is sequentially divided into smaller mutually exclusive subspaces by eliminating those points that cannot be the end points of the extrema.

Although the number of subspaces is increased two for each recursive call of the procedure, there are many more points that are eliminated from the search space after each call.

Therefore, the search space is rapidly decreasing. How rapid the decreasing rate is depends on the shape of R.

This method is derived from geometric considerations and consequently its rate of convergence is strongly dependent on the geometry of the region. It is, therefore, difficult to accurately access the computational complexity of the method. If used as a preprocessing technique, it operates in $0(4n)$ time, plus the time required to perform the convex hull calculation on the remaining points, or $0(k \log k)$ where k represents the points remaining after preprocessing.

Certainly in the worst case, almost no points are eliminated and convergence operates as convex hull in $0(n \log n)$. It is, in fact, worse than the convex hull in this case, since the program is more complex.

However, due to the large number of branch points, the algorithm exits early for virtually all regions, and it converges very rapidly.

SUMMARY

Four methods have been presented for finding the extrema of a region. The major axis approach is best if the "average" or "smoothed" shape of the region represents the desired property, and if spurs containing a small number of points can be ignored. If this is not the case, then one of the other methods is preferred.

If the convex hull will be needed for other analysis, then Method 2 is preferred.

Method 3 is probably the fastest and easiest to program, but it is not guaranteed to converge to the global maximum, although it does so with high likelihood.

Method 4 presents a technique which can be used in two ways, the first a preprocessing algorithm which dramatically reduces the search space before invoking a convex hull algorithm, and, second, a continuation which, although complex to program, generally converges quickly.

REFERENCES

[1] C. Brice and C. Fennama, "Scene analysis using regions," Stanford Res. Inst., Menlo Park, CA, Artificial Intelligence Group Tech. Memo, Nov. 17, 1970.

[2] R. T. Chien and W. E. Snyder, "Automatic visual inspection of hybrid circuits," in *Proc. 4th Int. Joint Conf. on Artificial Intelligence*, Tibilski, U.S.S.R., Summer 1976.

[3] R. Duda, J. Kremers, and D. Nitzan, "Automatic part classification," in *Exploratory Research in Advanced Automation*, 5th Rep., Stanford Res. Inst., Menlo Park, CA, 1975.

[4] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[5] R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Inform. Processing Lett.*, vol. 1, 1972.

[6] L. D. Harmon, "Automatic recognition of print and script," *Proc. IEEE*, vol. 60, pp. 1165-1176, 1972.

[7] R. Jarvis, "On the identification of the convex hull of a finite set of points in the plane," *Inform. Processing Lett.*, vol. 2, 1973.

[8] J. Munson, "Experiments in the recognition of hand-printed text," in *Proc. Fall Joint Comput. Conf.*, 1968.

[9] T. Pavlidis, "A review of algorithms for shape analysis," *Comput. Graphics Image Processing*, vol. 7, 1978.

[10] —, *Structural Pattern Recognition*. New York: Springer-Verlag, 1977.

[11] M. Shamos, "Geometric complexity," in *Proc. 7th ACM Symp. Theory of Computing*, 1975.

[12] Y. Yakimovsky and J. Feldman, "A semantics-based decision theory region analyzer," in *Proc. 3rd Int. Joint Conf. on Artificial Intelligence*, Stanford, CA, 1973.

High-Speed Multidimensional Convolution

CHANG EUN KIM AND MICHAEL G. STRINTZIS

Abstract—**Fast computation algorithms are developed for two-dimensional and general multidimensional convolutions. Two basic techniques (overlap-and-add, overlap-and-save) are described in detail. These techniques allow speed and storage requirement tradeoffs and they define a decomposition of the total convolution into partial convolutions that can be easily found by parallel use of fast sequential cyclic convolution algorithms. It is shown that unlike what is the case in one dimension, the "overlap-and-save" method enjoys a clear advantage over the "overlap-and-add" method with respect to speed and storage in multidimensional convolution. A specific computational burden is assessed for the case where these methods are used in conjunction with radix-2 fast Fourier transform algorithms.**

Index Terms—**Algorithms and computation analysis, convolution, image processing, multidimensional signals, pattern recognition.**

I. INTRODUCTION

The need for multidimensional convolution arises often and in many application areas. For example, such convolutions are commonly required in problems occurring in image processing and recognition of two- or three-dimensional features. Also, convolutions of higher dimension are used in areas such as tomography and in abstract classification procedures. However, as the dimension of the convolution increases, speed and storage computer requirements increase geometrically. In this paper, we describe a parallel-and-fast-sequential computation methodology that alleviates this problem.

Fast algorithms [1]-[6] for the computation of discrete Fourier transforms have been successfully used for fast computation of one-dimensional convolutions and correlations. In addition to the flexible FFT algorithm, several more efficient algorithms [8]-[10] have been developed for the specific purpose of high-speed convolution.

Probably the most well-known basic techniques for high-speed convolution are: 1) the "overlap-and-add" technique, and 2) the "overlap-and-save" technique [7], [11]. Although it has been clear [12] that these techniques are extendable to two or more dimensions, no precise definition of the extension or numerical evaluation of its results has been given until the recent preliminary report in [14] (for the two-dimensional case) and the present paper.

Two algorithms for two-dimensional convolution are developed in Section II. The numerical efficiency and flexibility of these algorithms are discussed in Section III. In Section IV, the multidimensional high-speed convolution method is presented and analyzed in its general setting. Discussion of the results appears in Section V.

II. BASIC ALGORITHMS

Consider the two-dimensional discrete convolution defined by

$$y(m,n) = \sum_{l=0}^{N-1} \sum_{k=0}^{M-1} h(k,l)\, x(m-k, n-l)$$

$$m, n = \cdots -1, 0, 1 \cdots \quad (1)$$

Manuscript received June 6, 1978; revised August 1, 1979.

The authors are with the Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, PA 15261.

0162-8828/80/0500-0269$00.75