Scale-Based Description and Recognition of Planar Curves and Two-Dimensional Shapes

FARZIN MOKHTARIAN AND ALAN MACKWORTH

Abstract—**The problem of finding a description, at varying levels of detail, for planar curves and matching two such descriptions is posed and solved in this paper. A number of necessary criteria are imposed on any candidate solution method. Path-based Gaussian smoothing techniques are applied to the curve to find zeros of curvature at varying levels of detail. The result is the "generalized scale space" image of a planar curve which is invariant under rotation, uniform scaling and translation of the curve. These properties make the scale space image suitable for matching. The matching algorithm is a modification of the uniform cost algorithm and finds the lowest cost match of contours in the scale space images. It is argued that this is preferable to matching in a so-called stable scale of the curve because no such scale may exist for a given curve. This technique is applied to register a Landsat satellite image of the Strait of Georgia, B.C. (manually corrected for skew) to a map containing the shorelines of an overlapping area.**

Index Terms—**Cartography, computational vision, curve recognition, generalized scale space, map generalization, path-based Gaussian smoothing, remote sensing, shape description, uniform cost algorithm, zeros of curvature.**

I. Introduction

COMPUTATIONAL vision systems interpret images of two-dimensional or three-dimensional objects. Two-dimensional shapes (such as letters of the alphabet, chromosomes and shorelines in satellite images are bounded by, or composed of, planar curves. Various curve representations have been proposed in the computational vision literature. A reliable representation, suitable for matching, should be essentially invariant with the rotation, scaling, and translation of the curve to make recognition of the curve possible after arbitrary instances of those transformations. Moreover, it should represent the curve at varying levels of detail rather than at only one level.

The goals of this paper are:

1) To find a method of obtaining a representation for a two-dimensional curve which is invariant under rotation, scaling, and translation of the curve.

2) To develop a matching algorithm to find the best match of two such representations. Such an algorithm should also be able to match one representation to part of another, to account for incomplete data.

3) To apply the techniques developed to satisfy 1) and 2) to register a Landsat satellite image of an area to a map which contains the shorelines of an overlapping area.

Manuscript received October 8, 1984; revised July 25, 1985. Recommended for acceptance by W. E. L. Grimson. This work was supported by the Natural Sciences and Engineering Research Council and the Canadian Institute for Advanced Research.

The authors are with the Laboratory for Computational Vision, Department of Computer Science, University of British Columbia, Vancouver, B.C. V6T 1W5, Canada.

IEEE Log Number 8405788.

II. Some Criteria for a Reliable Representation

A number of necessary criteria that any reliable method for curve description and recognition must satisfy are presented here.

a) The representation must be computable efficiently.

b) The representation should be essentially invariant under rotation, uniform scaling, and translation of the curve; otherwise, reliable recognition will not be possible.

c) The representation should contain information about the curve at varying levels of detail. Moreover, it should be clear from the representation which features of the curve belong to coarser levels of detail and which features to finer levels

d) The amount of change in the representation should correspond to the amount of change made to the curve. In other words, a small change to part of the curve should create a small local change in the representation.

e) Arbitrary choices should not affect the representation.

f) In case of open curves intersecting the frame, the representation should only change locally with the location of the cutoff points.

g) The representation should uniquely specify a single curve, otherwise it would be possible to match a curve against a class of curves all of which have the same representation. This criterion only requires uniqueness up to the curve equivalence classes induced by requirement b) above.

Several shape representation techniques may be judged by these criteria. For example, the Hough transform has been used to detect straight lines [1], circles [2], and arbitrary shapes [3] in images. O'Gorman and Clowes [4] used the direction of the gradient to improve the efficiency and accuracy of the transform. In a typical Hough transform technique, "edge" elements in the image vote for the parameters of the objects they could be located on. All the votes are registered in a parameter space. The highest peaks indicate the location of the objects in the image. The Hough transform can suffer from false peaks in the accumulator array due to accidental matches with the data. Poor results will be obtained for incomplete data even if the existing data matches well with the model. In order to account for rotation, scaling, and translation of the shape,

more parameters must be added to the parameter space which makes the implementation impractical. The generalized Hough transform [3] also suffers from the arbitrary choice of the required reference point for the shape.

Chain encoding [5], [6] and polygonal approximations [7], [8] have been used as representations for signals and curves. These representations do not usually reflect considerations of invariance b) or detail c).

Shape factors and quantitative measurements [9]–[11] have been used to describe shapes. These methods involve a substantial reduction in data and do not meet requirement g) as well as others.

Another class of methods are those which find hierarchical straight line or angle approximations to the curve [12]–[15]. A hierarchy of straight line approximations to the curve is formed by initially joining the endpoints and recursively refining each approximation by breaking it at the point on the curve furthest away from it. The algorithm stops when every point on the curve is within some threshold distance from the straight line segment which approximates the portion of the curve containing the point. This method does combine information about the curve at various levels of detail but the representation can change greatly due to a small change in the curve. Moreover, if the curve is closed, an arbitrary choice of endpoints must be made which will certainly affect the representation. Strip trees also use a similar idea in order to represent a curve. The only difference is that a rectangle which contains all of the points on a segment is used to represent that segment. Smaller and smaller rectangles are used to give a finer representation of the curve. The deficiencies of the previous method also make this one unattractive.

The "codon" representation proposed by Hoffman and Richards [16] satisfies many of our criteria in that it is based on segmenting at minima of curvature. However, it does not reflect considerations of detail c) or sensitivity d).

The "curvature primal sketch" introduced by Asada and Brady [17] also satisfies many of our criteria. However, they use only a limited number of well-defined shapes which can be approximated well by analytical functions.

A final candidate method breaks the curve into several segments (if needed) such that each segment is a single-valued function $y = y(x)$. Then the Stansfield–Witkin method [18], [19] can be used to construct a scale space image of the curve. The effectiveness of this method depends on the number of segments the curve has to be divided into. In the special case where the curve already is a single-valued function, no divisions are needed and the method would work but in general (when divisions are necessary), there will be problems with handling the boundary conditions at each break in the curve. This method violates criterion b) in that the representation is not invariant under rotation.

III. Scale-Based Description of Planar Curves

This section describes a method for computing a representation of a curve invariant under rotation, uniform scaling, and translation of the curve as explained in [20]. This method is based on finding points of inflection on the curve at varying levels of detail using a path length parameter and combining them to obtain the scale space image of the curve.

A. Finding Points of Inflection on a Planar Curve

It is desired to find zero-crossings in curvature of the curve at varying levels of detail, that is, for varying degrees of smoothing of the curve. Since a planar curve does not behave like a single-valued function in general, a parametrization of the curve should be found which makes it possible to compute the curvature of the curve at varying levels of detail. Such a parametrization is made possible by considering a path length variable along the curve and expressing the curve in terms of two functions $x(t)$ and $y(t)$:

$$C = \{x(t), y(t)\}$$

where t is a linear function of the path length ranging over the closed interval [0, 1]. If the curve is closed, $x(t)$ and y(t) are periodic functions. The curvature κ of a planar curve at a point P on the curve is defined as the instantaneous rate of change of the slope angle ψ of the tangent at point P with respect to arc length s, and is equal to the inverse of the radius ρ of the *circle of curvature* at point P.

$$\kappa = \frac{d\psi}{ds} = \frac{1}{\rho}$$

The circle of curvature at point P is a circle tangent to the curve at point P whose center lies on the concave side of the curve and whose curvature is the same as that of the curve at point P. The circle of curvature is also called the *osculating circle* because it has a higher degree of contact with the curve at point P than any other circle.

κ can be computed if it is expressed in terms of the derivatives of functions $x(t)$ and $y(t)$.

Define

$$y' = \frac{dy}{dx} \qquad y'' = \frac{d^2y}{dx^2}$$

It is known that

$$\kappa = \frac{y''}{(1 + (y')^2)^{3/2}}.$$

Therefore, y' and y'' should be expressed in terms of the first and second derivatives of $x(t)$ and $y(t)$.

Denote

$$\dot{x} = \frac{dx}{dt} \qquad \ddot{x} = \frac{d^2x}{dt^2} \qquad \dot{y} = \frac{dy}{dt} \qquad \ddot{y} = \frac{d^2y}{dt^2}$$

Then

$$y' = \frac{\dot{y}}{\dot{x}} \qquad y'' = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\dot{x}^3}$$

and

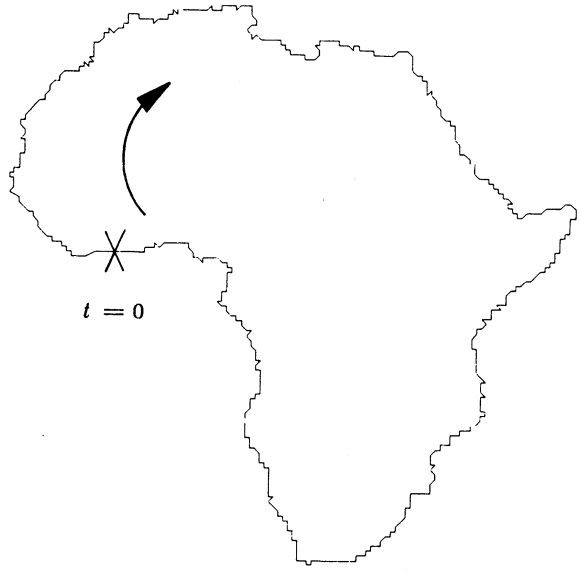


Fig. 1. Shoreline of Africa

$$\kappa = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{3/2}}.$$

In order to compute the curvature of the curve at varying levels of detail, functions $x(t)$ and $y(t)$ are convolved with a one-dimensional Gaussian kernel $g(t, \sigma)$ of width σ [21]:

$$g(t, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-t^2/2\sigma^2}.$$

$X(t, \sigma)$, the convolution of $x(t)$ and the Gaussian kernel, is defined as:

$$X(t, \sigma) = x(t) \circledast g(t, \sigma) = \int_{-\infty}^{\infty} x(u) \frac{1}{\sigma\sqrt{2\pi}} e^{-(t-u)^2/2\sigma^2} du.$$

$Y(t, \sigma)$ is defined similarly.

In the smooth curve one needs $\dot{X}(t, \sigma)$, $\dot{Y}(t, \sigma)$, $\ddot{X}(t, \sigma)$ and $\ddot{Y}(t, \sigma)$ to compute curvature. These can be computed from $x(t)$ and $y(t)$ using

$$\dot{X}(t, \sigma) = \frac{\partial X(t, \sigma)}{\partial t} = \frac{\partial [x(t) \circledast g(t, \sigma)]}{\partial t} = x(t) \circledast \left(\frac{\partial g(t, \sigma)}{\partial t}\right)$$

and

$$\ddot{X}(t, \sigma) = \frac{\partial^2 X}{\partial t^2} = x(t) \circledast \left(\frac{\partial^2 g(t, \sigma)}{\partial t^2}\right)$$

and similarly for $Y(t, \sigma)$. Fig. 1 shows the coastline of Africa. Fig. 2 shows an application of this method to that coastline, with the zeros of curvature marked on each smoothed curve.

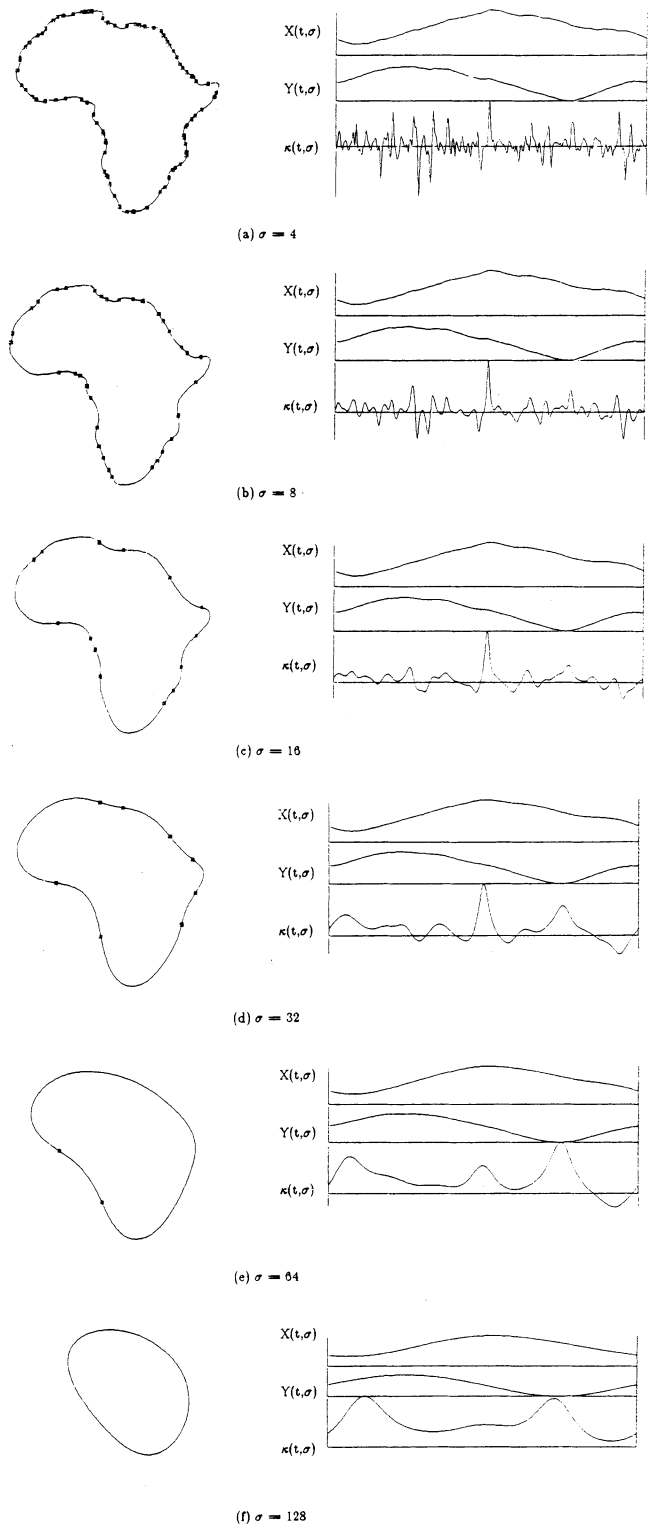


Fig. 2. Smoothing a curve: scale-based effects.

B. How to Handle the Endpoint and Cutoff Point Problems

If the curve is closed, functions $x(t)$ and $y(t)$ can be treated as periodic functions, which eliminates all edge effects. But if the curve is open, these functions are not periodic. A way must be found to convolve the mask with the curve when part of the mask is beyond the endpoint

since the convolution value for all points on the curve should be computed.

There is a basic difference in the nature of two open curves, one of which is contained completely inside the image and the other which ends at the frame boundary. In the first case there is in fact no missing information and the endpoint problem can be solved by creating an extension to the curve which is an extrapolation of points close to the endpoint. As long as one is consistent, the results for similar curves will be similar. In the second case there is indeed a loss of information where the curve is cut off by the frame boundary. We call this a cutoff point. It should be noted that the cutoff point is not a true endpoint of the curve and therefore is not perceptually important. Furthermore, it is possible to try to match two basically similar curves which have different cutoff points. There are several ways to handle this problem. Each one will be discussed.

a) One could still extrapolate the curve beyond the cutoff points. Since this does not in general result in a reconstruction of the original shape, the results can be different for two similar curves which are cut at different locations by the frame boundary.

b) The cutoff points of the curve can be joined with a straight line to change it into a closed curve. The arguments of section a) also apply here.

c) When the cutoff point of the curve is reached, turn back and go in the opposite direction, towards the other endpoint. This method will also not guarantee similar results and it also suffers from the fact that artificial inflection points will be introduced at the endpoints of the curve.

d) Repeat the cutoff point of the curve as many times as necessary: treat $x(t)$ and $y(t)$ as constant functions beyond the cutoff point. This method does not introduce extra inflection points and seems like the appropriate thing to do in absence of other information. It is also the easiest method to implement and is the one used in this paper to handle the endpoints as well as the cutoff points.

C. The Scale Space Image of a Planar Curve

Section III-A described how to find points of inflection on the curve at varying levels of detail. This section will describe how to combine that information in the form of a generalized scale space image in order to obtain a representation for the curve.

Compute a Gaussian mask using a value of σ corresponding to the finest level of detail desired. The limit is $\sigma = 0$ which is equivalent to convolving an impulse function with the functions $x(t)$ and $y(t)$. In practice a higher value can be used to avoid the excessive detail usually obtained at very fine levels of detail. Find points of inflection on the curve corresponding to this value of σ using the techniques explained in Section III-A. Mark these points in a coordinate system where the x-axis shows the value of the path-length parameter t along the curve and the y-axis represents σ, the width of the Gaussian kernel.

Increase the value of σ by a small amount, find the new inflection points and mark them also in the scale space image. Repeat this process until no inflection points are found for some value of σ. The whole scale space image has been derived (Fig. 3).

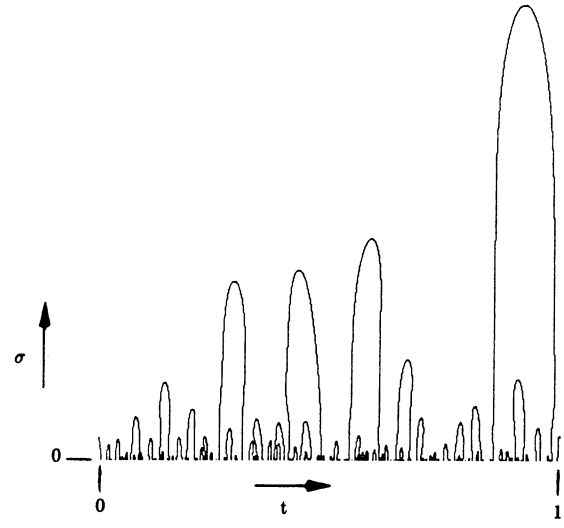


Fig. 3. Generalized scale space image of Africa.

The scale space image can be thought of as a binary image which is 1 wherever there is an inflection point on the curve corresponding to that pair of values of t and σ and 0 elsewhere.

Some remarks about the scale space image are appropriate.

a) The scale space image contains contours which are closed at the top and open at the bottom. There may be contours open at the top but these contours are incomplete ones which end at the edge of the scale space image and are due to incomplete information.

b) It can be shown that contours in a scale space image can intersect each other [22]. However, this is very unlikely and we have made the assumption that it does not happen in our application.

c) Yuille and Poggio [23] have shown that under certain assumptions, no new inflection points are created at higher levels of detail (more smoothing). This property can be used to speed up the computation of the scale space image by a considerable amount. Indeed once one is beyond fine levels of detail, one can track the inflection points by only computing the curvature and looking for zero-crossings in the neighborhood of the previous inflection points.

d) Yuille and Poggio [24] have also shown that almost all signals can be reconstructed up to an equivalence class from their scale space images. This implies that the scale space image is a unique representation for those signals.

e) For some open curves, there may be one or more incomplete contours in their scale space images. Since the "movement" of such contours towards the edge can be quite slow, a lot of unnecessary computation time will be spent if such a contour is allowed to take its normal course. A shortcut can be made by anticipating cases such as this. As soon as only one inflection point is obtained on the whole curve for some value of sigma, find the slope of the corresponding contour to determine whether it is moving

toward the right or the left edge and connect the contour to the correct edge.

IV. An Algorithm to Match Scale Space Images

The matching is carried out in the scale space image because of the invariance properties it displays. Furthermore, matching in the scale space image is preferable to matching at a specific scale. A specific scale is usually chosen by imposing some stability criterion on the features of the curve. There does not always exist a "most stable" scale for a given curve. That approach would be unreliable for such a curve.

We proceed with a number of definitions. A scale space image is considered to be a hierarchical representation for a curve. An imaginary super-contour exists at the root of this tree. This contour contains all the real contours in the scale space image. Each real contour has zero or more children, contours which exist inside it, and zero or more siblings, contours which share the same parent. The *peak* of a contour is defined to be the highest point on that contour. The *right branch* of a contour consists of all the connected points on the contour starting to the right of the peak. Similarly, the *left branch* of a contour consists of all the points on the contour starting to the left of the peak. To *transform* a contour is to apply a scale space transform to it. As a result, the contour may be shifted horizontally and/or scaled uniformly. The average distance between two contours is the average of the distances between the peaks, the right branches, and the left branches. The cost of matching two contours is defined to be the average distance between them after one of them has been transformed.

The algorithm described in this section is an adaptation of the Uniform Cost Algorithm [25] which is a special case of the A^* algorithm [26]. Its goal is to find the best match of contours in a scale space image to some or all of the contours in another scale-space image. The image space transformation parameters which take one curve to the other are then computed from that match. There are four parameters which correspond to uniform scaling, rotation, and translation of the curve. The two curves can then be registered to show the goodness of match between them.

The following is a step-by-step description of the matching algorithm.

1) Create a number of initial nodes corresponding to the possible match of every pair of contours in the two scale space images.

2) For each node created in step 1, compute two scale space transformation parameters. If there were a perfect match, applying the transformation to the smaller contour would make it identical to the larger contour.

3) Compute the cost of match for each node created in step 1.

4) Remove the lowest cost node from the queue and expand it into a new node. A unique pair of contours, one contour from each scale space image, is found. These two must be a match for each other if the previous pair were in fact a correct match. Compute the cost of match for the new pair and add it to the previous cost. Add the new node to the queue.

5) Repeat step 4 until the lowest cost node can not be expanded further, i.e., there are no more contours in either of the two scale space images left to match for that node. Return that node as the best match of the two scale space images and stop.

The following is a detailed explanation of node creation and expansion.

A. Creating the Initial Matching Nodes

The algorithm starts by creating a queue of nodes corresponding to the possible match of every pair of contours in the two scale space images. Since the two original curves could be at different scales, it is possible for contours at different levels to match with each other. There are two scale space transformation parameters mapping one contour to another which need to be computed for each node. These two are the scale k and shift d parameters. The relationship between the old coordinates, t and σ, and the new coordinates, t' and σ', is as follows:

$$t' = kt + d$$

$$\sigma' = k\sigma$$

Only one pair of scale space points is needed to compute k and d. These can be computed using the coordinates of the peaks of the two contours since peaks correspond to each other.

These parameters are always used to transform the smaller contour so that it can be matched against the larger one. As a result, contours do not shrink and matching errors are reduced. The same set of parameters is used to match the next pairs of contours when a node is expanded.

No nodes are created for incomplete contours since their true shape is not known. Since it is desirable to find a match corresponding to the coarse features of the curves, there is a penalty associated with starting a match with a small contour. This penalty is a linear function of the difference in height of that contour and the tallest contour of the same scale space image and is added to the cost of match computed when a node is created.

B. Expanding a Node

In order to expand a node, the next pair of contours in the scale space images to be matched must be found. Instead of randomly choosing any other contour in the scale space image and finding the next match value, this algorithm makes use of the constraints of the scale space image as much as possible.

To find the next two contours to match, a contour is searched for in the first scale space image using an order-by-height procedure. If it is found, than a contour is searched for (but not necessarily found) in the second image corresponding to the contour in the first image. The following is a detailed description of this procedure. Note that if at any step the program fails to find a contour from

the first image, it will search for a contour in the second image using the same guidelines for finding contours in the first image before going to the next step.

a) If the last contour matched from the first scale space image has any children, then its tallest child will be the next contour to be matched. In order to find the next contour in the second image to match, use the transformation parameters to estimate its location. Since its parent is known, one can search for any children of that parent whose peaks fall inside that range. If more than one such contour is found, choose the one whose height is closest to that estimated by the transformation parameters. If no corresponding contour is found in the second image, the first contour will be matched against the null contour, which has a height of zero.

b) If the last contour matched from the first image does not have any children, search for its next smaller sibling. If such a sibling exists, it will be the next contour to be matched. To find the next contour from the second image, use the procedure outlined in a).

c) If the last contour matched from the first image does not have any smaller siblings either, return one level and search for a smaller sibling of its parent. Repeat this process until either the first ancestor which has a smaller sibling is reached or there are no more ancestors left. In the latter case, no more expansion is possible and the algorithm returns with the lowest cost node.

If the curve for which the scale space image is computed is closed, then the scale space image will be periodic: moving right when one is at the right edge of the scale space image will place one at the left edge and *vice versa*. This must be taken into account when applying transformations to the contours since contours may wrap-around in the scale space image.

C. Dealing with Incomplete Information

If one of the curves being matched only matches with part of the other curve, the contours in the scale space image for that curve will correspond to only some of the contours in the other scale space image. This will not result in poor matches since it is easy to find a region in the larger scale space image which corresponds to the smaller scale space image (again wrap-around is possible for closed curves) and extract the relevant contours which should be used in the matching process.

V. Application

The Landsat registration problem was chosen to illustrate the curve description and recognition methods explained in Sections III and IV. The Landsat registration problem is that of registering a Landsat satellite image of an area to a map which contains the shorelines of the same area. Shorelines must be extracted from the Landsat image before they can be matched against the shorelines in the map. It should be noted that exact registration of the shorelines is not a goal of this paper. The purpose is to demonstrate that the method is successful in finding shapes which have basically the same features.

Fig. 4. Band 7 Landsat image of part of Strait of Georgia, B.C.

Section V-A explains a simple method for extracting shorelines from the Landsat image. Section V-B describes a way of putting together the curve description and recognition methods in order to create a working system to solve the Landsat registration problem.

A. Obtaining Shorelines from the Landsat Image

The Band 7 (near infrared) Landsat image of part of the Strait of Georgia, B.C., is a gray-level image showing a clear distinction between water and land in most areas (Fig. 4). A histogram of gray-level distributions of all the pixels in the Landsat image was obtained [Fig. 5(a)]. There are two main peaks in this histogram corresponding to water and land, respectively, but there are also many smaller peaks due to noise.

The minimum value of σ, the standard deviation of a Gaussian, which results in two peaks can be obtained using binary search. The resulting smoothed histogram is shown in Fig. 5(b). To find the location of the peaks, one can simply convolve the first derivative of the Gaussian, using the correct value of σ, with the histogram and look for transitions from positive to negative. The trough between the two peaks can similarly be found by looking for a transition from negative to positive.

Once the location of the trough is known, the Landsat image can be thresholded using the intensity value of the trough as the thresholding value. Every pixel in the image will be classified as water if its gray-level is less than the thresholding value and as land if its gray-level is greater than or equal to the thresholding value. The result is a binary image where 1 pixels represent land and 0 pixels represent water.

To obtain the land–water boundaries from the thresholded image, one need only follow the contours of the land masses. One simple algorithm for following the contours of light regions in binary images is given in [27]. Fig. 6 shows the approximations to the shorelines extracted from the Landsat image after manual correction for skew. For a more detailed explanation see [28].

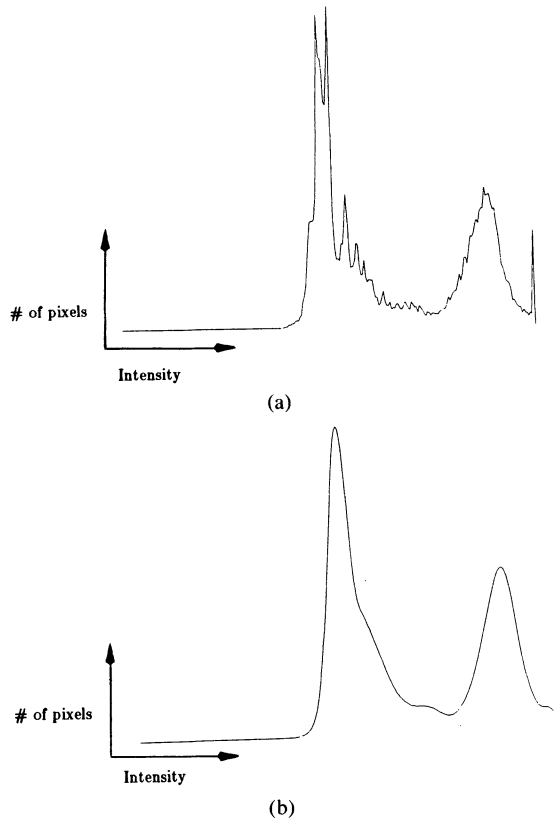


Fig. 5. Gray-level distribution of pixels in the Landsat image. (a) Original histogram. (b) Smoothed histogram with two peaks.

Fig. 6. Shorelines from the Landsat image after correction for skew.

B. A Working System

The complete working system to register the Landsat image and the map is as follows.

a) Threshold the Landsat image using the value of the trough between the land and water peaks in the smoothed histogram of the gray-level distribution of the pixels in the image. Follow the contours of dark regions to find land–water boundaries.

b) Eliminate as much of the skew in the shorelines from the Landsat image as possible. This is done manually by finding corresponding pairs of points in the Landsat image and a map (different from the one used in part c) and using them to estimate the parameters of an affine transformation which takes the Landsat image to the map. A transformation of uniform scaling, rotation and translation still exists between the Landsat image and the map used in part c.

c) Choose only a few contours from the Landsat image and the map which are longer than all the other contours. The purpose of this is to improve the efficiency of the program but also to eliminate very low cost matches corresponding to very small contours which may not have any significance. From these contours keep only the ones which are closed.

d) Compute the scale space images for the contours which were selected by the previous step and create a hierarchical representation based on that image.

e) Match every model curve against every object curve and remember the cost of match for all those pairings. Since every curve considered in the matching process has a significant length, the lowest cost match should also be a correct one. Find such a match, compute the transformation parameters predicted by that match and choose a subset of all the matches which are consistent with that match, in that they predict roughly the same transformation parameters. Use a least squares method to estimate the parameters of an affine transformation from the Landsat image to the map using the data gathered from all the correct matches.

f) Generate the transformed image of the Landsat shorelines and transform the original Landsat image if so desired.

VI. Results

The map to which the Landsat image was to be registered is shown in Fig. 7. It should be noted that the transformation between the Landsat image and the map is a general affine transform. Since currently only rotation, uniform scaling and translation can be handled by the matching algorithm, the skew in the Landsat image had to be corrected before the scale space images for its shorelines could be computed. Since a relatively small amount of skew still remains in the Landsat shorelines even after manual correction, it is necessary to compute the parameters of an affine transformation which takes the Landsat image to the map once a consistent subset of matches have been found. Figs. 8 and 9 show two correct matches found by the matching algorithm and Fig. 10 shows the Landsat image registered to the map using the same matching algorithm. The goodness of match between the Landsat image and the map indicates that a small number of shorelines matched correctly can give a good estimate for the transformation parameters.

It was stated in Section V that only closed curves were chosen for matching purposes. One reason for this is that the scale space images corresponding to closed curves are complete and it is preferable to use complete information

Fig. 7. Map of part of the Strait of Georgia, B.C.

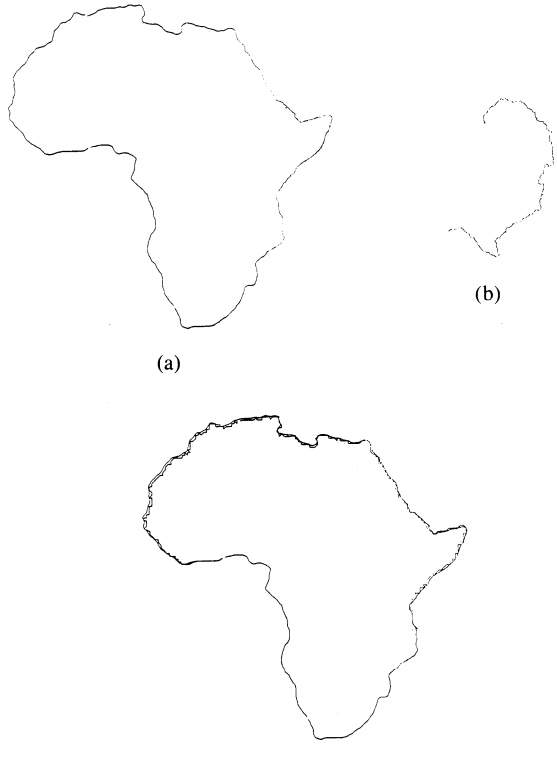


Fig. 8. Matching shorelines of Africa. (a) Complete shoreline of Africa. (b) Part of the same shoreline. (c) Result of the match.

when available. The other reason is a potential problem with the matching algorithm as described in Section IV when matching open or incomplete curves. It was stated that an initial cost is associated with every node when it is created which is a linear function of the difference in heights of the contour for which the node is created and the highest contour of the scale space image. This measure is to encourage matches corresponding to larger contours of the image and is reasonable when both images are complete since the correct match does correspond to the large contours of the image (usually the largest ones) but it can run into problems when there is incomplete information since a correct match might not correspond to one of the largest contours. The matching algorithm in its present form is good at finding shapes which have the same basic features but to overcome the problem associated with open curves, one can eliminate the necessity of attaching initial costs to nodes by matching all contours in the scale space image (including parents and ancestors and siblings of those) which should match as a result of matching any two contours in the two images (not just the smaller siblings and subcontours of them) and making sure that duplicate nodes are not created.

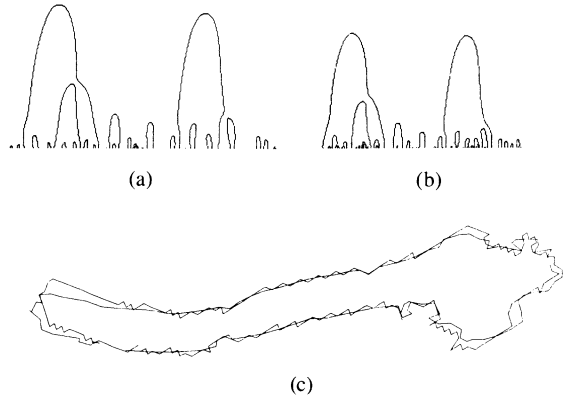


Fig. 9. Matching shorelines from the map and the Landsat image. (a) Scale space image of map island. (b) Scale space image of Landsat island. (c) Result of the match.

Fig. 10. Registration of the Landsat image to the map.

Since computing time is always an important constraint in every working computational vision system, the following is an evaluation of the computation time requirements of the various components of the system described in this paper.

The system was implemented at UBC's Laboratory for Computational Vision on a Vax[1] 11/780 with 4 Mbytes of main memory running BSD 4.2 Unix.[2] Those parts of the system concerned with extracting shorelines from the Landsat image and computing the scale space images of

[1]Vax is a trademark of Digital Equipment Corp.
[2]Unix is a trademark of AT& T Bell Laboratories.

the curves to be matched were implemented in C. The matching algorithm was implemented in Franz Lisp.

Tracking the zero-crossings in the scale space image can reduce the computation time required to find the contours from hours to minutes for a typical curve. The algorithm to match two scale space images, each with n contours in it, is $O(n^3)$ in the worst case. This is the case because the matching algorithm initially creates n^2 nodes. The first node, corresponding to the two root contours, can be expanded a maximum of $n - 1$ times. This figure goes down for the rest of the nodes and depends on where matching starts. However, the matching algorithm usually does better because many nodes are eliminated quite early due to high costs attached to them. Therefore a typical run is between $O(n^2)$ and $O(n^3)$ in CPU requirements. This argument is supported by experimental results. Matching two scale space images each with four to five contours took roughly 20 seconds and matching two scale space images each with approximately 20 contours took roughly 300 seconds.

If real-time response is expected from the system described in this chapter, its CPU requirements must be decreased further. Parallelism would significantly reduce the CPU requirements of the system described in this paper.

VII. Discussion

We can now evaluate the generalized scale-space image according to the criteria proposed in Section II. The scale space image can be computed efficiently therefore it satisfies criterion a). It is completely invariant under translation of the curve. Rotation of the curve can produce at most a wrap-around translation in the t dimension. Scaling of the curve corresponds to a translation in the σ dimension. So criterion b) is satisfied. The representation does combine information at varying levels of detail c) and it usually changes gradually as the shape of the curve is changed d). No arbitrary choices need to be made to compute the representation e) and changing the cutoff points only changes the representation locally f). Finally, criterion g) is usually satisfied since shorelines do have unique scale space images. This is obviously not true for convex shapes. Possible extensions to overcome this weakness are given in Section VIII.

Small changes in the shape of the curve usually result in small changes in its scale space image; however, this may not seem to be true if a relatively small but highly convex feature is added to the curve. Such a feature may last through many levels of scale despite its relatively small size and therefore create a relatively large contour in the scale image; but it can be argued that such a feature is easily distinguishable and hence a major feature of the curve. Fig. 11 illustrates this.

The whole scale space image of a curve is not more compact than the curve itself in general (this is obviously not true for convex or nearly convex curves). Therefore, the program which computes the scale space image of various curves usually produced fairly large binary images, but this should not be viewed as a shortcoming of the scale space image since it has value as a representation for planar curves. Furthermore, the scale space image could be efficiently encoded.

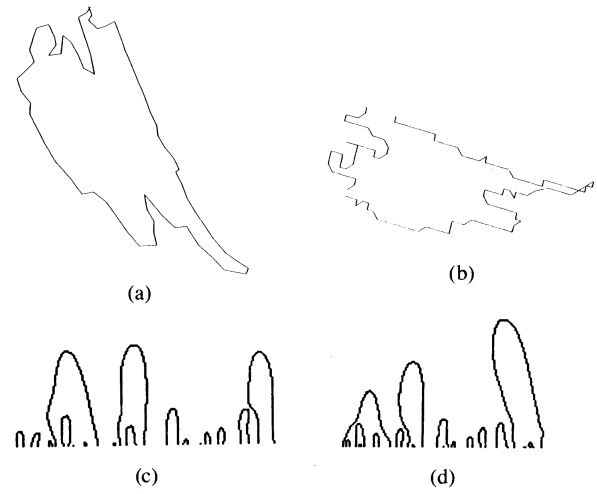


Fig. 11. Effects of features on the scale space image. (a) Island from the map. (b) Island from the Landsat image. (c) Scale space image of (a). (d) Scale space image of (b).

Shorelines have quite arbitrary shapes and rich scale space images as a result, but the scale space image may not be a suitable representation for curves which consist of long straight segments or curves which are convex or nearly convex to begin with.

VIII. Extensions

The shapes of the contours in the scale space image indicate certain facts about the underlying features on the curve and can be studied further. This is useful if no explicit model is available but certain shapes (cues) are searched for on the curve. For example, a contour which is relatively tall in the scale space image but has a small base (distance between the points where its left and right branches intersect the path-length axis of the scale space image), indicates the presence of a highly convex feature of limited extent.

Scale space images can be obtained for planar curves which consist of more than one segment by finding a suitable parameterization. They can also be computed for curves which intersect themselves, but it should be noted that the scale space images for this class of curves will be fundamentally different. It is, for example, possible to obtain new inflection points at coarser levels of detail.

If the effect of skewing of a planar curve on its scale space image could be determined, then the matching algorithm could be modified to detect skew in a planar curve also.

Once the Landsat image is registered to the map, the results can be used to improve the inital segmentation of the Landsat image. An edge detector can be used to find the location of land–water boundaries at various levels of scale.

The generalized scale space records only the sign of the curvature. Thus, for example, any two wholly convex

shapes are indistinguishable to the method. Various possible extensions could be considered to cope with this problem. Using the locations of curvature maxima and minima at varying scales would reduce the size of the equivalence classes but would not eliminate the difficulty. A second extension would follow or combine scale space matching with image space matching based on a Euclidean distance. The third (and the most satisfactory) possible extension requires generalizing the scale space image again to record not just the sign of curvature but also its magnitude. The matching algorithm and its associated cost metric would have to be extended to cope with this.

Last but not least, it is possible to extend these results to three dimensions by obtaining the three-dimensional scale space image of a surface and using a generalized contour matcher.

IX. Conclusions

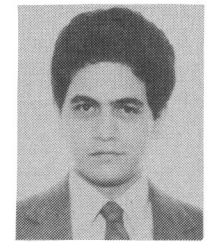
We have posed the problem of scale-based description of planar curves, proposed a number of criteria for any solution method, and described a method which more nearly satisfies those criteria than do the other candidate methods. We have also described a matching algorithm which optimally matches two such shape descriptions.

Acknowledgment

We are grateful to R. Woodham and J. Little for their comments.

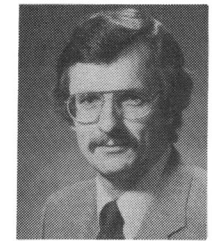
References

[1] P. V. C. Hough, "Method and means for recognizing complex patterns," U.S. Patent 3 069 654, 1962.
[2] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11-15, 1972.
[3] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111-122, 1981.
[4] F. O'Gorman and M. B. Clowes, "Finding picture edges through collinearity of feature points," in *Proc. 3rd Int. Joint Conf. Artificial Intell.*, Stanford, CA, 1973, pp. 543-555.
[5] H. Freeman, "Computer processing of line-drawing images," *Comput. Surveys*, vol. 6, no. 1, Mar. 1974.
[6] J. W. McKee and J. K. Aggarwal, "Computer recognition of partial views of curved objects," *IEEE Trans. Comput.*, vol. C-26, no. 8, pp. 790-800, 1977.
[7] T. Pavlidis, "Polygonal approximations by Newton's method," *IEEE Trans. Comput.*, vol. C-26, no. 8, pp. 800-807, 1977.
[8] ——, "Segmentation of pictures and maps through functional approximation," *CGIP*, vol. 1, pp. 360-372, 1972.
[9] P. E. Danielsson, "A new shape factor," *CGIP*, vol. 7, pp. 292-299, 1978.
[10] E. E. Underwood, *Quantitative Stereology*. Reading, MA: Addison-Wesley, 1970.
[11] I. T. Young, J. E. Walker, and J. E. Bowie, "An analysis technique for biological shape," *Inform. Contr.*, vol. 25, pp. 357-370, 1974.
[12] A. K. Mackworth, "On reading sketch maps," in *Proc. 5th Int. Joint Conf. Artificial Intell.*, Cambridge, MA, 1977, pp. 598-606.
[13] T. Pavlidis, *Structural Pattern Recognition*. New York: Springer-Verlag, 1977.
[14] L. S. Davis, "Understanding shape: Angles and sides," *IEEE Trans. Comput.*, vol. C-26, no. 3, pp. 236-242, 1977.
[15] D. H. Ballard and C. M. Brown, *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
[16] D. D. Hoffman and W. A. Richards, "Representing smooth plane curves for recognition: Implications for figure-ground reversal," in *Proc. Nat. Conf. Artificial Intell.*, Pittsburgh, PA, 1982, pp. 5-8.
[17] H. Asada and M. Brady, "The curvature primal sketch," M.I.T. AI Memo 758, 1984.
[18] J. L. Stansfield, "Conclusions from the commodity expert project," M.I.T. AI Memo 601, 1980.
[19] A. P. Witkin, "Scale space filtering," in *Proc. 8th Int. Joint Conf. Artificial Intell.*, Karlsruhe, West Germany, 1983, pp. 1019-1022.
[20] A. K. Mackworth and F. Mokhtarian, "Scale-based description of planar curves," in *Proc. 5th Canadian Soc. Computational Studies of Intell.*, London, Ont., Canada, May 1984, pp. 114-119.
730, 1983.
[21] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. Royal Soc. London B*, vol. 207, pp. 187-217, 1980.
[22] I. Katz and A. K. Mackworth, "Open contours and crossing contours in scale-space images of one dimensional functions," Dep. of Comput. Sci., Univ. British Columbia, Vancouver, B.C., Canada, Tech. Rep. (in preparation), 1985.
[23] A. L. Yuille and T. Poggio, "Scaling theorems for zero-crossings," M.I.T. AI Memo 722, 1983.
[24] ——, "Fingerprint theorems for zero-crossings," M.I.T. AI Memo 730, 1983.
[25] N. J. Nilsson, *Problem-Solving Methods in Artificial Intelligence*. New York: McGraw-Hill, 1971.
[26] ——, *Principles of Artificial Intelligence*. Palo Alto, CA: Tioga, 1980.
[27] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.
[28] F. Mokhtarian and A. Mackworth, "Scale-based description and recognition of planar curves and two-dimensional shapes," Dep. Comput. Sci., Univ. British Columbia, Vancouver, B.C., Canada, Tech.



Farzin Mokhtarian received the B.E.S. degree in mathematical sciences from the Johns Hopkins University, Baltimore, MD, in 1982, and the M.Sc. degree in computer science from the University of British Columbia, Vancouver, B.C., Canada, in 1984.

For six months in 1984 and 1985 he was an associate member of the Professional Staff at Schlumberger-Doll Research Laboratory in Ridgefield, CT, where he was involved in research in seismic image processing. He is currently enrolled in the Ph.D. program at the University of British Columbia. His research interests are in computational vision and artificial intelligence.



Alan Mackworth studied at the University of Toronto, Harvard University, and Sussex University.

He is currently a Professor of Computer Science at the University of British Columbia, Vancouver, B.C., Canada. He is also a Senior Fellow of the Canadian Institute for Advanced Research. Since his move to UBC, his research has focused on the theory of computational vision and applications in the interpretation of satellite imagery and geographical sketch maps. Results on the representation and use of knowledge in vision systems, including surface orientation representations, network consistency algorithms, and schema-based systems, are described in more than 30 scientific papers.

Dr. Mackworth has served as President of the Canadian Society for Computational Studies of Intelligence, and as the Chairman of the Board of Trustees of International Joint Conferences of Artificial Intelligence, Inc. He was the Director of the UBC Laboratory for Computational Vision from 1980 to 1984, when he became UBC Coordinator of the Canadian Institute for Advanced Research Program in Artificial Intelligence and Robotics.