SCALE-BASED DESCRIPTION AND RECOGNITION OF PLANAR CURVES

by

FARZIN MOKHTARIAN

B.E.S., Johns Hopkins University, 1982

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF COMPUTER SCIENCE

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

August, 1984

© Farzin Mokhtarian, 1984
In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of **COMPUTER SCIENCE**

The University of British Columbia
1956 Main Mall
Vancouver, Canada
V6T 1Y3

Date **August 23, 1984**
Abstract

A method of finding points of inflection on a planar curve at varying levels of detail and combining them to obtain a representation of the curve invariant under rotation, uniform scaling and translation and an algorithm to match two such representations are developed.

This technique is applied to register a Landsat aerial image of an area (corrected for skew) to a map containing the shorelines of the same area.

The shorelines are extracted from the Landsat image by forming a histogram of the gray-level distribution of pixels in the image and finding the land and water peaks in that histogram. The value at the trough between the two peaks is then used to threshold the image. Contours of dark regions in the resulting binary image are taken to be the shorelines.

The Uniform Cost algorithm is used to find the least cost matches of the "scale-space images" of shorelines extracted from the Landsat image and those in the map giving priority to matches at higher levels of scale. A subset of those matches which are consistent are chosen to estimate the parameters of an affine transformation from the image to the map using a least squares method.
# Table of Contents

1. Introduction ................................................................. 1  
   1.1 Area of research .................................................. 1  
   1.2 Reading Guide ..................................................... 2  

2. Representation techniques ............................................... 5  
   2.1 Some criteria for a reliable representation ................. 5  
   2.2 Survey of techniques .......................................... 6  
      2.2.1 Hough transform ........................................ 6  
      2.2.2 Straight line approximations .......................... 9  
      2.2.3 Chain codes ........................................... 10  
      2.2.4 The $\psi - s$ curve ................................ 11  
      2.2.5 Other methods .......................................... 11  

3. Curve description and recognition ................................... 14  
   3.1 Scale-Based description of planar curves ................... 14  
      3.1.1 Finding points of inflection on a planar curve .... 14  
      3.1.2 How to handle the endpoints of the curve .......... 17  
      3.1.3 The scale-space image of a planar curve ........... 21  
   3.2 An algorithm to match scale-space images .................. 24  
      3.2.1 Creating initial nodes ................................ 25  
      3.2.2 Expanding a node ..................................... 26  
      3.2.3 Dealing with incomplete information ................. 27  
   3.3 Obtaining shorelines from the Landsat image ................ 28  
      3.3.1 Forming a Histogram .................................. 28  
      3.3.2 Locating land and water peaks ....................... 31  
      3.3.3 Thresholding the image ............................... 33  
      3.3.4 A simple contour follower ............................. 33  

4. Implementation ............................................................ 36  
   4.1 A working system ............................................... 36  
   4.2 Computer systems ............................................... 37  
   4.3 A collection of programs ...................................... 37  
   4.4 Programming Languages ........................................ 38  
   4.5 Input and output to and from the programs .................. 38  
   4.6 Program descriptions ......................................... 39  
      4.6.1 Curve ................................................... 41  
      4.6.2 Contours ............................................... 42  
      4.6.3 2dmatch ................................................. 43
5. Results and Conclusions .................................................................................................................. 45
  5.1 Results ........................................................................................................................................ 45
    5.1.1 Efficiency .......................................................................................................................... 52
  5.2 Conclusions .................................................................................................................................. 53
    5.2.1 Future work ......................................................................................................................... 54

References ........................................................................................................................................... 57
Appendix 1. Estimating the transformation parameters ................................................................. 59
Appendix 2. Manual pages of a few programs in the public library .............................................. 61
List of Figures

Figure 2.1 The normal parameters for a line ............................................. 6
Figure 2.2 Generalized Hough transform for an arbitrary shape .................. 6
Figure 2.3 $\psi - \sigma$ segmentation .......................................................... 11
Figure 3.1 Shoreline of Africa with the starting point ............................... 18
Figure 3.2 Smoothing a curve: Scale-based effects .................................... 19
Figure 3.3 Scale-space image of Africa ..................................................... 23
Figure 3.4 Band 7 Landsat image of part of the strait of Georgia, B.C. .......... 29
Figure 3.5 Histogram of gray-level distribution of pixels in Landsat image .... 30
Figure 3.6 Convolving the histogram with a Gaussian function .................. 32
Figure 3.7 Thresholded Landsat image ..................................................... 34
Figure 5.1 Map of part of the Strait of Georgia, B.C. ................................. 46
Figure 5.2 Shorelines extracted from the Landsat image ............................ 47
Figure 5.3 Shorelines from the Landsat image after correction for skew .......... 48
Figure 5.4 Matching shorelines of Africa .................................................. 49
Figure 5.5 Matching islands from the map and the Landsat image ............... 50
Figure 5.6 Registration of the Landsat image to the map ......................... 51
Figure 5.7 Effects of features on the scale-space image ............................. 55
Acknowledgements

I am very grateful to my supervisor, Alan Mackworth, for showing care and interest in my work and for being generous with his time when I needed guidance. I am also grateful to Bob Woodham for his helpful comments on an earlier draft of this thesis.

Many thanks go to Jim Little, Jan Mulder and Alan Carter for helpful discussions and to my family for their continued support.

Last but not least, “Thank you” to Marian for giving me her “unconditional” love and for fighting it with me.
CHAPTER 1

Introduction

1.1. Area of research

The research described in this thesis falls into the area of computational vision or image understanding. Computational vision is a sub-discipline of artificial intelligence which is concerned with reaching an understanding or interpretation of an input image. The image is often a two-dimensional array of values where each value gives the intensity or color of light at that point in the image (grey level or color image), but it could also be a line drawing in which case it is assumed that the lines have been extracted from an image by another process or created directly.

The basic difference between computational vision or image understanding and image processing is that there is no understanding (even in a limited sense) of the input data in image processing. Both the input and output data to and from an image processing program are images. The output image is obtained by applying a transformation to the input image. Blurring, sharpening, adding noise, removing noise, enhancement, and restoration (Rosenfeld and Kak, 1976) are examples of image processing tasks. The output from an image understanding program is some more explicit information or knowledge about the image. This information may become the input to a higher level process or be the information needed about the image.

Images being studied by computational vision can be those of 2 or 3 dimensional objects. Two dimensional objects (such as letters of the alphabet,
geometric figures and shorelines in aerial photographs) usually consist of one or more planar (two dimensional) curves. Various methods for finding representations for two dimensional curves have been proposed and exist in computational vision literature. A reliable representation should be invariant with the rotation, scaling and translation of the curve to make recognition of the curve possible under arbitrary amounts of the above transformations. Moreover, it should represent the curve at varying levels of detail rather than at only one specific level.

The goals of this thesis are:

(a) Find a method of obtaining a representation for a two dimensional curve which would be invariant under rotation, scaling and translation of the curve, and

(b) 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 (in case of incomplete data).

(c) Apply the techniques developed in (a) and (b) to register a Landsat aerial image of an area to a map which contains the shorelines of the same area.

1.2. Reading Guide

Chapter 2 explains some of the previous work done on finding representations of two dimensional curves. A number of criteria for such representations will be proposed and each method will be evaluated according to those criteria.

Chapter 3 explains the method developed for finding a representation of a two dimensional curve which is invariant with the rotation, scaling and
translation of the curve. The curve is expressed as two distinct functions of a path length parameter and its curvature expressed in terms of the first and second derivatives of these functions. A Gaussian filter is convolved with the curve at varying levels of detail and zeroes of curvature extracted. The result is a scale-space image of the curve which is used as the representation.

This chapter also describes the matching algorithm which is based on the $A^*$ algorithm. This algorithm matches the contours in the scale-space image in a coarse to fine order and uses height to find the order in which contours will be matched. Closed curves are matched satisfactorily using this algorithm. Modifications to the algorithm to obtain better results for incomplete data are discussed in chapter 5.

A goal of this thesis has been to apply the techniques just mentioned to the Landsat registration problem. Therefore a method is needed to extract shorelines from the Landsat image. This method is also covered in chapter 3.

Chapter 4 explains the implementation of the theory described in chapter 3. The first part of the theory concerned with obtaining the scale-space image has been implemented using the C language. The matching algorithm has been implemented in Franz LISP.

Chapter 5 discusses the results obtained by implementing the theory and evaluates the method according to the criteria introduced in chapter 2. Directions for further work are also discussed.

Appendix 1 explains how the 4 transformation parameters are computed by the matching algorithm once a match is found.

Appendix 2 contains the manual pages for a few programs developed during the course of this thesis which are available on-line on the Laboratory for
Computational Vision's Vax 11/780 at the University of British Columbia.
CHAPTER 2

Representation techniques

2.1. Some criteria for a reliable representation

A desirable representation for a 2 dimensional curve would be one which can be relied on to make recognition possible given arbitrary amounts of rotation, scaling and translation of the curve. A number of criteria that a reliable method should satisfy will be presented here:

(a) The representation must be computable.
(b) The representation should be invariant under rotation, uniform scaling and translation of the curve, otherwise, reliable recognition will not be possible.
(c) The representation should consist of information about the curve at varying levels of detail. Moreover, it should be clear from the representation, which features of the curve belong to a coarser level of detail and which features to finer levels.
(d) The amount of change in the representation should correspond directly to the amount of change made to the curve. In other words, a small change to any part of the curve should create a small change in the representation with the amount of change in the representation becoming larger as the curve becomes more and more different.
(e) Arbitrary choices should not affect the representation.
(f) In case of open curves cut by the frame boundary, the representation should only change locally with 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.

2.2. Survey of techniques

A number of different methods for computing representations of 2 dimensional curves will be discussed in this section and each one will be evaluated according to the criteria mentioned in the previous section.

2.2.1. Hough Transform

Hough Transform has been used to detect straight lines (Hough, 1962; Duda and Hart, 1972) in images. In this method, every figure point is transformed into a straight line in a parameter space. The parameters used to describe lines in the picture plane define the parameter space. Hough chose to use the slope-intercept parameters of a line. Duda and Hart have used a different parametrization to avoid the unboundedness of the slope and intercept. This parametrization is illustrated in Figure 2.1. $\theta$ is the angle that the normal to the line makes with the x-axis and $\rho$ is its distance from the origin. The equation of the line in terms of these parameters is:

$$x \cos \theta + y \sin \theta = \rho. \quad (2.1)$$

There are an infinite number of lines which go through a single point in a plane. In practice, the parameter space is quantized into a number of subspaces or cells. A cell gets a vote when the parameters of a line which goes through any point in the image plane fall inside the ranges represented by that cell. When all the votes have been cast, the cells with the highest number of votes give the parameters of the most likely lines in the image. O’Gorman and Clowes (1973) used the direc-
Figure 2.1. The normal parameters for a line.

Figure 2.2. Generalized Hough transform for an arbitrary shape.
tion of the gradient to improve the efficiency and accuracy of the transform. The gradient should be perpendicular to the direction of the edge element. This fact can be used to reduce the number of possible lines through an edge element significantly.

Duda and Hart also extended the Hough Transform to detect circles in images. In their extension, the parameter space becomes 3 dimensional because of the 3 parameters needed to uniquely specify a circle in a plane. The direction of the gradient at the image point was used by Kimme, Ballard and Sklansky (1975) in order to improve the efficiency of the circle finding algorithm of Duda and Hart.

Ballard (1981) generalized the Hough transform to detect arbitrary two dimensional shapes. He has pointed out that analytical shapes (such as circles, parabolas and ellipses) can be detected by a Hough transform technique which takes every image point to an n dimensional parameter space where n is the number of parameters it takes to uniquely specify that analytical shape. If rotation and scaling of the shape are allowed, extra parameters must be added to the parameter space to account for them. Arbitrary shapes are handled as follows:

Choose a reference point y for the shape (Figure 2.2). For each point x on the boundary of the shape, compute \( \phi(x) \) the gradient direction and \( r = y - x \). Store r as a function of \( \phi \). A table will be constructed this way which in general gives several values of r for each value of \( \phi \). For each edge element x in the picture, compute \( \phi \), the gradient direction and increment all the points \( x + r \) in an accumulator array where r is indexed by \( \phi \) in the table constructed before.

O'Rourke (1983) observed that the quantization of parameter space into equal size cells can run into a serious storage problem when the space has several dimensions. He has proposed the use of a Dynamically Quantized Space. In a DQ
space, cells can be split or merged according to the local characteristics of the data. A cell is split if its count or its internal imbalance is too high. Imbalance is simply a function of the distribution of hits inside the boundaries of a cell. Similarly, two cells are merged if the sum of their counts is not too large and the resulting cell is not highly imbalanced.

All Hough transform techniques completely skip the problem of finding a suitable representation. They all attempt to recognize a shape by using simply the coordinates of edge points in the image (or gradient direction at most). Therefore the only representation is the coordinates of points and/or gradient direction. Coordinates obviously change with rotation, translation and scaling of the shape. The gradient direction changes with rotation only. There is no representation of the curve at different levels of detail. Moreover, the Hough transform can suffer from false peaks in the accumulator array due to accidental match of 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 also suffers from the arbitrary choice of the required reference point for the shape.

2.2.2. Straight line approximations

The next class of algorithms are those which find straight line approximations to the curve (Mackworth, 1977; Ballard and Brown, 1982). A hierarchy of straight line approximations to the curve is formed by initially joining the endpoints and recursively refining each approximation by breaking the previous one at a point on the curve furthest away from it. The algorithm stops when every point on the curve is within some distance from the straight line segment which
approximates that portion of the curve containing the point.

This method does combine information about the curve at various levels of detail but the representation could 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 same deficiencies of the previous method also make this one unattractive.

Davis (1977) has observed that regions of high curvature in a curve are important during the recognition process and has proposed a hierarchical angle detection scheme which proceeds by finding the angles between line segments joining a point to its neighboring points on the curve. Local maximums are found and marked as points of high curvature on the curve. This process continues by considering a larger neighborhood of points and searching for local maxima in curvature.

This method can also run into problems if a small change is made to the curve. Curvature is not actually computed but approximated using the angle between lines joining points in some neighborhood. If several angles are competing for a local maximum, a small change in the shape could change the outcome of the competition, giving a representation which is different from the one obtained before the change was made.

2.2.3. Chain codes

Chain codes (Freeman 1974) are obtained by representing the starting
point on the curve by its coordinates and coding the successive points by their displacements (on an imposed grid) from the previous point. The derivative of a chain code is another sequence of numbers indicating the relative change in the direction of a chain segment with respect to the previous segment. Chain codes are specially vulnerable to small changes in the shape and do not combine information at varying levels of detail. Moreover, the choice of the starting point will result in a shift in the chain code. The amount of that shift can not be determined easily in general.

2.2.4 The \( \psi-s \) Curve

The \( \psi-s \) curve (Ambler et al. 1975) can be obtained by plotting \( \psi \), the angle between a fixed line and a tangent to the boundary of a curve, against \( s \), the length of the path along the curve (Figure 2.3). Closed curves will yield a function which is periodic.

The \( \psi-s \) curve also suffers from the arbitrary choice of a required fixed line, and there is no combination of information at varying levels of detail.

2.2.5 Other methods

One possible method is to break the curve into several pieces (if needed) such that every piece is a single-valued function. Then Witkin's method (1983) can be used to construct a scale-space image of the curve. The effectiveness of this method depends on the number of pieces the curve has to be divided into such that every piece would be a single-valued function. In the special case where the curve already is a single-valued function, no divisions will be needed and the method will work but in general (when divisions are necessary), there will be problems with handling the boundary conditions at each break in the curve. The resulting representation could be poor if several breaks have been made in the
Figure 2.3. $\psi-s$ segmentation. (a) Planar curve and a tangent. (b) $\psi-s$ curve.
curve.

Another method calls for approximating the curve using polynomials. The parameters of the polynomials provide a representation for the curve. The fact that the parameters of the approximating polynomials would change under rotation, scaling or translation of the curve makes this approach undesirable also.
CHAPTER 3

Curve description and recognition

Section 3.1 explains a method of computing a description of a planar curve invariant under rotation, uniform scaling and translation of the curve. This description combines features of the curve at varying levels of detail and can be used to match a curve against another one.

An algorithm for matching two descriptions (scale-space images) of two different curves is explained in section 3.2. This algorithm is based on the uniform cost algorithm and favors matches at higher levels of scale.

The Landsat registration problem is chosen to illustrate these techniques. Shorelines must be extracted from the Landsat image before these techniques can be applied. A simple method to extract shorelines from the Landsat image is described in section 3.3.

3.1. Scale-Based description of planar curves

This section describes a method for computing a representation for a curve invariant under rotation, uniform scaling and translation of the curve. 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.

3.1.1. Finding points of inflection on a planar curve

It is desired to find zero-crossings in curvature of the curve at varying lev-
els of detail, that is, for varying levels of smoothness (or roughness) 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. If the curve is closed, $x(t)$ and $y(t)$ will be periodic functions. The curvature $\kappa$ of a planar curve at a point $P$ on the curve is defined as the instantaneous rate of change of the slope angle $\psi$ at point $P$ with respect to arc length $s$ (Figure 2.3a), and is equal to the inverse of the radius $\rho$ 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 has the same curvature as 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.

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

Define

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

It is known that:
\[ \kappa = \frac{y'''}{\left(1+(y')^2\right)^{3/2}} \]

therefore \( y' \) and \( y'' \) should be expressed in terms of the first and second derivatives of \( x(t) \) and \( y(t) \).

Denote

\[
\begin{align*}
\dot{x} &= \frac{dx}{dt} \\
\ddot{x} &= \frac{d^2x}{dt^2} \\
\dot{y} &= \frac{dy}{dt} \\
\ddot{y} &= \frac{d^2y}{dt^2}
\end{align*}
\]

Then

\[ y' = \frac{\dot{y}}{\dot{x}} \]

and

\[ y''' = \frac{d}{dx} \left( \frac{\dot{y}}{\dot{x}} \right) = \frac{\frac{d\dot{y}}{dt} \left( \frac{\dot{y}}{\dot{x}} \right)}{\dot{x}} = \frac{\dot{x} \dddot{y} - \ddot{y} \dddot{x}}{\dot{x}^3} \]

using the derivative rule:

\[
\left( \frac{u}{v} \right)' = \frac{u'v - v'u}{v^2}
\]

we obtain

\[ \kappa = \frac{\dot{x} \dddot{y} - \ddot{y} \dddot{x}}{(\dot{x}^2 + \dddot{y}^2)^{3/2}} \]

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 \( G(t) \) of width \( \sigma \) (Marr and Hildreth 1980):

\[ G(t) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{t^2}{2\sigma^2}} \]

\( X(t,\sigma) \), the convolution of \( x(t) \) and the Gaussian kernel, is defined as:
\[ X(t, \sigma) = x(t) \otimes g(t, \sigma) = \int_{-\infty}^{\infty} x(u) \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(t-u)^2}{2\sigma^2}} \, du \]

\[ Y(t, \sigma) \] 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

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

and

\[ \ddot{X}(t, \sigma) = \frac{\partial^2 X}{\partial t^2} = x(t) \otimes \left( \frac{\partial^2 g(t, \sigma)}{\partial t^2} \right) \]

and similarly for \( Y(t, \sigma) \). Figure 3.1 shows the coastline of Africa. Figure 3.2 shows an application of this method to that coastline.

3.1.2. How to handle the endpoints of the curve

As mentioned in the previous chapter, if the curve is closed, functions \( x(t) \) and \( y(t) \) can be treated as periodic functions which will eliminate all edge effects. But if the curve is open, these functions won’t be 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 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 being consistent, the results for similar curves will be similar. In the second case there is indeed a
Figure 3.1. Shoreline of Africa with the starting point
Figure 3.2. Smoothing a Curve: Scale-based effects.
Figure 3.2. (Continued) Smoothing a Curve: Scale-based effects.
loss of information around the endpoint. There are several ways to handle this problem. Each one will be discussed:

(a) One could still extrapolate the curve beyond the endpoints. Since this won't 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 endpoints 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 endpoint of the curve is reached, turn back and go in the opposite direction, i.e., towards the other endpoint. This method will also not guarantee similar results but it also suffers from the fact that artificial inflection points will be introduced at the endpoints of the curve.

(d) Repeat the endpoint of the curve as many times as necessary, i.e., treat \(x(t)\) and \(y(t)\) as constant functions beyond the endpoint. This method won't 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 thesis to handle the endpoints.

3.1.3. The scale-space image of a planar curve

Section 3.1.1 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 scale-space image (Witkin 1983) in order to obtain a representation for the curve.

Compute a Gaussian mask using a value of \(\sigma\) 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 \( \sigma \) using the techniques explained in section 3.2.1. Mark these points in a coordinate system where the x-axis shows the value of the path-length parameter along the curve and the y-axis represents \( \sigma \), the width of the Gaussian kernel.

Increase the value of \( \sigma \) by a small amount every time, 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 \( \sigma \). The whole scale-space image has been derived (Figure 3.3).

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 some values of the path-length parameter and sigma and 0 otherwise.

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) Contours in the scale-space image never intersect each other but may "touch" at fine levels of detail.

(c) Yuille and Poggio (1983) 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 passed 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.
Figure 3.3. Scale-space image of Africa
Yuille and Poggio (1983) have also shown that almost all signals can be reconstructed from their scale-space images. This implies that the scale-space image is a unique representation for those signals.

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 towards the right or the left edge and connect the contour to the correct edge using a straight line.

3.2. An algorithm to match scale-space images

A scale-space image can be 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, i.e., contours which exist inside it, and zero or more siblings, i.e., contours which share the same parent. Every contour has a peak, a right branch and a left branch unless it is incomplete in which case one branch will be totally missing.

The algorithm described in this chapter is an adaptation of the Uniform Cost Algorithm (Nilsson 1971) which is a special case of the $A^*$ algorithm (Nilsson 1980). It finds the lowest cost match between contours in the two scale-space images (see section 3.2.1 for a definition of the cost function). The transformation parameters which take one curve to the other are then computed from that match. There are 4 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.

3.2.1. Creating initial nodes

The algorithm starts out by creating a number of nodes corresponding to the possible match of every two contours in the two scale-space images. Since the two original curves can exist at different levels of scale, it is possible for contours at different levels to match with each other. There are two 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 $\sigma$, and the new coordinates, $t'$ and $\sigma'$, is as follows:

$$
\begin{align*}
t' &= k \cdot t + d \\
\sigma' &= k \cdot \sigma
\end{align*}
$$

Only one pair of points is needed to compute $k$ and $d$. These can be computed using the coordinates of the peaks of the two contours since peaks provide a pair of points on the contours which correspond to each other.

These parameters are used to transform the smaller contour (regardless of whether it is a model or an object contour) so that it can be matched against the larger one. The same set of parameters is used to match the next pairs of contours when a node is expanded.

The cost of match between two contours is defined as the average distance between them once one of them has been transformed. The average distance between two contours is the average of the distances between the peaks, the right branches and the left branches. Incomplete contours are not matched since their true shape is not known. Since it is desirable to find a match corresponding to
the basic or 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 scale-space image.

3.2.2. Expanding a node

The algorithm proceeds by finding the lowest cost node from the queue of initial nodes and expanding it. The cost of expanding that node is computed and added to the previous cost. Then the new node is added to the queue and this process is repeated until the lowest cost node can not be expanded any more. The correct match is assumed to be the one indicated by this node.

In order to expand a node, the next two 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 model scale-space image using an order-by-height procedure. If it is found, then a contour is searched for (but not necessarily found) in the object scale-space image which would correspond to the model contour. The following is a detailed description of this procedure. Note that if at any step the program fails to find a model contour, it will search for an object contour using the same guidelines for finding model contours before going to the next step.

(a) If the last contour matched from the model scale-space image has any children then its tallest child will be the next model contour to be matched. In order to find the next object contour 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 there are more than one such contours found, choose the one whose height is closest to what is estimated by the transformation parameters. If no corresponding object contour is found, the model contour will be matched against the null contour (which has a zero height).

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

(c) If the last model contour matched does not have any smaller siblings either, go back 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 (the starting point is reached). 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, i.e., 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 it is possible for a contour to wrap-around in the scale-space image.

3.2.3. 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.

3.3. Obtaining shorelines from the Landsat image

The Landsat image (taken in band 7 to emphasize the difference between water and land) is a gray-level image which shows a clear distinction between water and land in most areas (Figure 3.4). Since there is usually a sharp difference in gray-level between adjacent water and land pixels, one may be tempted to use an edge detector to locate water-land boundaries. It was decided not to use an edge detector because an edge detector is not biased in favor of land-water boundaries. Using an edge detector one would obtain many edges which do not correspond to boundaries of land and water. Moreover, an edge detector may produce an edge which corresponds partly to a land-water boundary and partly to a brightness discontinuity inside land or water. The method described in this section does show the desired bias in favor of land-water boundaries and does not produce many irrelevant edges (those which don’t correspond to land-water boundaries).

3.3.1. Forming a Histogram

A histogram of gray-level distributions of all the pixels in the Landsat image can be obtained (Figure 3.5). There are two main peaks in this histogram corresponding to water and land respectively but there are also smaller peaks due to noise and one at the very end of the histogram corresponding to saturated pixels (snow patches). If one could locate the land and water peaks on the histogram, the value of the trough between those two peaks could be used to threshold the image. The contours of dark regions in the thresholded image could then be followed to obtain approximations to the shorelines.
Figure 3.4. Band 7 Landsat image of part of the Strait of Georgia, B.C.
Figure 3.5. Histogram of gray-level distribution of pixels in Landsat image
### 3.3.2. Locating land and water peaks

These peaks can in fact be found by convolving the histogram $h(t)$, which can be treated as a function of one variable, with a mask based on a one-dimensional Gaussian function $G(x)$ (Glicksman 1982) where:

$$G(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

and the convolution is expressed as:

$$H(t,\sigma) = h(t) \ast G(t,\sigma) = \int_{-\infty}^{\infty} x(u) \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(t-u)^2}{2\sigma^2}} du$$

The effect of the convolution is to replace each pixel by a weighted average of pixels in a certain neighborhood of that pixel with the weight of a pixel decreasing as it gets further away from the center of the mask. The size of the neighborhood is controlled by $\sigma$, the parameter of the Gaussian (Figure 3.6).

The minimum value of $\sigma$ which results in two peaks can be obtained using binary search. To find the location of the peaks one can simply convolve the first derivative of the Gaussian:

$$G'(x) = -\frac{x}{\sigma \sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

using the correct value of $\sigma$ with the histogram and look for transitions from positive to negative (since the first derivative measures the slope of the smooth curve and a peak is where slope goes from a positive value to a negative value). Similarly the trough between the two peaks can be found by looking for a transition from negative to positive.
Figure 3.6. Convolving the histogram with a Gaussian function
3.3.3. Thresholding the image

Once the value of the trough is known, the Landsat image can be thresholded using the 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 (Figure 3.7).

The land and water pixels are assumed to be normally distributed with the mean of the distribution at the location of the peak. Since normal distributions only go to zero at infinity, there is some overlap of the two distributions. The intersection point of the land and water distributions, when chosen as a thresholding value to threshold the image, can be shown to result in a minimum number of land and water pixels classified incorrectly. A thresholding value larger than this value will result in fewer water pixels classified incorrectly but more land pixels classified incorrectly. Similarly, a thresholding value less than this value will result in fewer land pixels classified incorrectly but more water pixels classified incorrectly.

It was decided to use the value at the trough to threshold the image since the algorithm to find the trough was easier to implement and it was not believed that the difference in results would be significant for the purpose of this thesis.

3.3.4. A simple contour follower

To obtain the land-water boundaries from the thresholded image, one only needs to follow the contours of the land masses. One simple algorithm for following the contours of light regions in binary images (Duda and Hart, 1973) starts from any point on the contour and turns left if it is on a light pixel and right if it
Figure 3.7. Thresholded Landsat image
is on a dark pixel. This process is repeated until the whole region is traversed and one of the neighbors of the starting point is reached. As a result the contour of the region will be traversed in a clockwise fashion. If it is desired to traverse the contour of the region in a counter-clockwise fashion, one should turn right when on a light pixel and left when on a dark pixel. This may be necessary in order to traverse the contour of a region only partly contained in the image.

The authors have indicated that their algorithm may show 4-connected or 8-connected behavior depending on the starting point and the direction in which it is entered. One can easily force consistency on this algorithm by finding all the light pixels in the image which are only connected diagonally and disconnecting them. All the resulting regions will be 4-connected therefore the algorithm will always show 4-connected behavior also. Disconnecting pixels which are only connected diagonally is not an undesirable effect since two such pixels don't have any common boundary and can be considered as disconnected.
CHAPTER 4

Implementation

4.1. A working system

This section describes one way of putting together the curve description and recognition methods explained in chapter 3 in order to create a working system to solve the Landsat registration problem:

(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. See section 5.1 for some reasons.
(d) Compute the scale-space image 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, i.e., they predict roughly the same transformation parameters. This method might fail if all of the matches are poor ones or if small contours are also allowed in the matching process. An alternative is to use a Hough transform technique to let all matches "vote" for the correct 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.

4.2. Computer systems

The programs which realized the underlying theory were implemented on a VAX 11/780. Implementation started using UNIX® 4.1 BSD Operating System and was completed using UNIX 4.2 BSD Operating system.

4.3. A collection of programs

It was decided during the implementation stage that this thesis should be implemented as a collection of programs independent from each other, each one performing a task logically separate from the others. The obvious advantage is

† UNIX is a Trademark of Bell Laboratories.
that any of the programs can be run, tested, debugged and modified without having to touch any of the other programs in the same collection. Programs should have well-defined input and output so that it is easy to make one program interact with another. Moreover, these programs can be available as public programs on the computer system on which they were implemented. Any other user of the system who is working on a topic which requires similar tasks to be carried out, might be able to use one or more of the programs in that collection.

4.4. Programming Languages

The programming languages C and LISP were used to implement all of the programs which realize this thesis. C was chosen for programs which search for shorelines in the Landsat image, build the scale-space image and build a representation using the scale-space image. Since a large portion of the computation time is spent doing convolutions (building the scale-space image), it seemed that C would provide the desired efficiency. It is also a language which is well-supported by UNIX.

Since the matching algorithm used has a more symbolic flavor, it was implemented in Franz LISP, a dialect of LISP running on UNIX.

4.5. Input and output to and from the programs

Every program has its own defined input and output but if all of the programs in the collection are considered together, then the input to this collection of programs is a Landsat image and a sketch map of the same area from which the image was taken and the output from this collection is the shorelines found in the Landsat image brought into registration with the shorelines in the sketch map. The following is a description of various kinds of input and output structures used for the programs in the collection:
Plot Files: A plot file is a LISP list in which every element is of the form:

(*plot x y)

or

(*goto x y)

where *plot simply means join (pen down) the previous point to the current one (specified by its coordinates x and y) and *goto (pen up) means move to the current point (it signifies the beginning of a new chain of points). Any line drawing can be completely specified using such a format.

Image Files: An image file consists of a header which gives information about the number of rows and columns of the image, the number of bits used to encode each pixel, whether it is signed or not, and whether it is positive or not, and the body which holds the values of pixels in encoded form. Various routines to perform different operations on image files were already available in the LCV Standard Image System.

Functions and Histograms: These are just special kinds of image files in which there is only one row. The important difference is that the values of pixels don’t represent the gray-level of the image at that pixel, but the value of the function or the height of the histogram at that point.

4.6. Program descriptions

The following is a list of all the programs with a description of each program:

Chains This program implements a simple contour follower. The input is a binary image (256 x 256) and the output in a plot file containing the boundaries of the 'black' regions in the image.

Contours Creates a representation based on zero-crossing contours extracted
from a curve. Input is a plot file representation of the curve and the scale-space image of that curve which is a binary image and the output is a hierarchical representation of the contours in that scale-space image containing information about each contour. This representation is a LISP list and is used by the matching program.

**Curve**

Performs several functions. It can convolve a curve with a Gaussian based filter obtaining a 'smoother' version of the curve, show the zero-crossings in curvature on that smooth curve, show the \( x,y \) and \( \kappa \) (curvature) as functions of path-length and create the scale-space image of the curve.

**Curves**

Simply calls *Curve* followed by *Contours* in a loop to obtain representation for a number of chains and puts them together. The result is a collection of representations for all of the curves in a sketch map or all of the curves (shorelines) found in a Landsat image.

**Histogram**

Constructs a histogram of the grey-level distributions of the pixels in an image.

**Mean-Var**

Finds the peaks and Variances of the two peaks found in a histogram after convolving it with a Gaussian mask based on an appropriate value of \( \sigma \).

**Peak**

Finds the locations of the peaks and troughs in a histogram convolved with the first derivative of the Gaussian.

**Smooth**

Finds a value for \( \sigma \) such that when a Gaussian filter based on that value of \( \sigma \) is convolved with an input histogram, a required number of peaks is obtained.

**Threshold**

Thresholds images using one or two thresholding values. If two values are given, it can find the intersection or the 'inverse
intersection' of the two thresholded images. It can also classify the pixels into 3 groups based on whether the value of the pixel is less than the first value, between the two values or larger than the second value.

**2dmatch** Takes representations for the curves in a sketch map and those found in a Landsat image, finds the best matches for each curve, estimates the transformation parameters based on the best matches and uses those parameters to register the Landsat shorelines to those in the sketch map.

Appendix 2 contains manual pages for programs *curve*, *chains* (renamed as *contours*), *smooth*, and *peak*. These manual pages are available on-line on the LCV's Vax computer.

Of the ten programs just mentioned, *curve*, *contours* and *2dmatch* are the more substantial ones and require more detailed descriptions:

**4.6.1. Curve**

The scale-space image created by *Curve* is a binary image which shows the locations of zero-crossings with respect to path-length for varying values of $\sigma$. To obtain the scale-space image, a Gaussian mask based on $\sigma = 1.0$ is convolved with the curve and locations of zero-crossings found. The value of $\sigma$ is then increased by a small amount (0.2) and the new filter is convolved with the curve. In order to increase the efficiency of the program, the contours are tracked by convolving the filter with the curve only in the neighborhood of previous zero-crossings. This decreases the CPU requirements of the program significantly.

Gaps can appear in contours when the change in the slope of the contour is large. This happens specially at the peak of the contour and it is an undesir-
able feature since a contour must be connected everywhere in order to be recognized correctly by Contours. One solution is to use very small increments for $\sigma$ but this method runs into serious problems with CPU and storage requirements. The solution adopted in Curve was to fill such gaps. A gap is filled by simply joining two loose ends (a loose end is a 1 pixel in the scale-space image which has only one 1 neighbor, excluding the pixels in the first row) which are in the same neighborhood. When a loose end is discovered, search for another loose end starts in a very small neighborhood (two pixels away) and the neighborhood is gradually expanded if the other loose end is not found. Search is abandoned if a maximum size neighborhood is reached but no loose ends found.

4.6.2. Contours

The following information about each contour in the scale-space image is computed and written to the representation:

- **x-peak**: The height of the peak of the contour.
- **y-peak**: The value of the path-length where the peak of the contour happens to be.
- **x-ref**: The x-coordinate of a point on the original curve where the peak of the contour is located.
- **y-ref**: The y-coordinate of that point. This information is used by 2dmatch to compute the transformation parameters after the best match has been found.
- **partial**: True if the contour is partial, i.e., not all of the contour exists in the scale-space image. This happens as a result of missing information.
- **use-less**: True if contour is partial and its parent (the one above it) is also partial.
parent The number of the contour on top of this contour. If no such contour exists, super-contour is assumed to be the parent (contour number 1).

children The contours which are inside this contour.

y-min The minimum value of path-length that any point on the contour has (it must be a point on the left branch of the contour).

y-max The maximum value of path-length that any point on the contour has (it must be a point on the right branch of the contour).

x-rb The heights of the points on the right branch of the contour as traveled from the peak to the end of that branch.

y-rb The path-length values of the points on the right branch of the contour.

x-lb The heights of the points on the left branch of the contour.

y-lb The path-length values of the points on the left branch.

4.6.3. 2dmatch

Program 2dmatch initially creates a queue of nodes which correspond to all the possible matches between two representations for two curves. A match value is computed for each node and always the node which has the lowest cost of match of all is removed from the queue, expanded one step and added to the queue again. This process is repeated until no further expansion is possible. See chapter 3 for more details about the matching algorithm.

A node carries certain information which makes it unique and makes it possible to expand that node if it is selected. That information is the following:

contours The numbers of the two contours (one from the model, one from the object) whose possible match is being considered.

match-value The cost of match for this node so far.

parameters The parameters that take contours in the object representation to
the contours in the model representation or vice versa. These are
different from the transformation parameters which take the object
curve to the model curve.

\( \text{cn}\)\text{um} \quad \text{The number of the contour pairs that have been matched inside this node so far.}

\( \text{mp}\)\text{rev-path} \quad \text{The path from the model root contour to the last model contour which was matched the previous time this node was expanded.}

\( \text{op}\)\text{rev-path} \quad \text{The path from the object root contour to the last object contour which was matched the previous time this node was expanded.}

\( \text{mc}\text{-relevant} \) \quad \text{The relevant model contours. Only these contours from the model representation will be matched against the relevant object contours from the object representation for this node.}

\( \text{oc}\text{-relevant} \) \quad \text{The relevant object contours.}
CHAPTER 5

Results and Conclusions

This chapter discusses the results obtained by implementing the theory as described in chapter 4. The method is evaluated according to the criteria proposed in chapter 2. The chapter ends by discussing the strong and weak points of the generalized scale-space image and outlining possible paths for future work on the scale-space image.

5.1. Results

Figures 5.1 and 5.2 show the map and the shorelines extracted from the Landsat image respectively. 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. Figure 5.3 shows the Landsat shorelines after an attempt was made to correct the skew. This was done manually by finding 3 pairs of corresponding points in the Landsat image and the map. 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. Figures 5.4 and 5.5 show two correct matches found by the matching algorithm and figure 5.6 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
Figure 5.1. Map of part of the Strait of Georgia, B.C.
Figure 5.2. Shorelines extracted from the Landsat image
Figure 5.3. Shorelines from the Landsat image after correction for skew
Figure 5.4. Matching shorelines. (a) Complete shoreline of Africa. (b) Part of the same shoreline. (c) Result of the match.
Figure 5.5. Matching islands from the map and the Landsat image. (a) Scale-space image of island from the map. (b) That of island from the Landsat image. (c) Result of the match.
Figure 5.6. Registration of the Landsat image to the map.
parameters.

It was stated in chapter 4 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 when available. The other reason is a potential problem with the matching algorithm as described in chapter 3 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 adequate to give a relatively good (by visual inspection) registration between the Landsat image and the map 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.

5.1.1. Efficiency

Since time is always an important constraint in every working vision system, the following is an evaluation of the computation time requirements of the various components of the system described in this thesis:
Tracking the zero-crossings in the scale-space image can reduce the required computation time from hours to minutes for a typical curve. It took approximately one hour of CPU time to compute the scale-space images of all the curves in the map and slightly less than that to compute those of the shorelines in the Landsat image. The matching algorithm took roughly 30 minutes of CPU time to estimate the transformation parameters. As mentioned earlier, only a subset of the scale-space images were used in the matching process.

If real-time response is expected from the system described in this chapter, its CPU requirements must be decreased further. Section 5.2.1 introduces parallelism as a possible way of significantly reducing CPU requirements of the system described in this thesis.

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 computed 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, binary images can be encoded efficiently.

5.2. Conclusions

A number of criteria for any representation method were proposed in chapter 2. The generalized scale-space image can now be evaluated according to those criteria:

1. The scale-space image of a curve is invariant under rotation and translation of the curve (modulo translation in the scale-space). It is also invariant under uniform scaling of the curve since it combines features about the curve at all levels of scale.
(2) The arbitrary choice of starting point for closed curves does not affect the representation.

(3) Small changes in the shape of the curve usually result in small changes in the representation. Although this may not 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-space image but it can be argued that such a feature is easily distinguishable and therefore a major feature of the curve. Figure 5.7 illustrates this.

(4) 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.

5.2.1. Future Work

Parallelism should be studied as a way to significantly reduce the required computation time for the system described in this thesis. The convolution of the initial histogram of the gray-level distribution of the pixels in the Landsat image with various masks can be speeded up as well as thresholding the image and following the contours of dark regions. Computation of the scale-space image can also be speeded up by "dividing" the work between as many processors as available. The matching algorithm can also benefit from a parallel architecture. The cost of match for every two curves in the map and the Landsat image can be computed in parallel. Applying transformations to contours in the scale-space images and finding their average distances is a time-consuming process which can also be done in parallel.

The shapes of the contours in the scale-space image indicate certain facts
Figure 5.7. Effects of features on the scale-space image. (a) Island from the map. (b) Island from the Landsat image. (c) & (d) Scale-space images of (a) and (b) respectively.
about the underlying features on the curve and can be studied further. This is useful if no explicit model curve is available but certain shapes (cues) are searched for on the curve. For example, a contour which is relatively large 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), possibly indicates the presence of a highly convex feature which is small in size.

Scale-space images can be obtained for planar curves which consist of more than one segment by finding a suitable parametrization. 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 higher levels of scale.

Skewing of a planar curve translates to skewing of its scale-space image. Therefore the matching algorithm can be modified to detect skew in a planar curve. What is needed is to find at least one more pair of corresponding points on the contours in the scale-space image (in addition to the peaks) in order to estimate the skew parameters.

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

Last but not least, it is possible to extend these results to three dimensions by obtaining the scale-space image of a surface (which will be two dimensional) and using a two dimensional contour matcher.
REFERENCES


Appendix 1

Estimating the transformation parameters

It was assumed in chapter 3 that the transformation which takes the image to the map consists of uniform scaling, rotation and translation. Such a transformation can be represented as:

\[ X' = R \cdot S \cdot X + T \]

where

\[ X = \begin{pmatrix} x \\ y \end{pmatrix}, \quad X' = \begin{pmatrix} x' \\ y' \end{pmatrix} \]

are the coordinates of a point before and after the transformation respectively,

\[ R = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \]

represents counter-clockwise rotation by an angle \( \theta \),

\[ S = \begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix} \]

represents uniform scaling by a factor \( s \), and

\[ T = \begin{pmatrix} x_t \\ y_t \end{pmatrix} \]

represents a translation of \( x_t \) in the \( x \) direction and \( y_t \) in the \( y \) direction.

Solving for \( x' \) and \( y' \) in terms of \( x \), \( y \) and the transformation parameters yields:

\[ x' = s \cos \theta x - s \sin \theta y + x_t \]  \hspace{1cm} (A2.1)

\[ y' = s \sin \theta x + s \cos \theta y + y_t \]  \hspace{1cm} (A2.2)

Equations (A2.1) and (A2.2) can be rewritten as:

\[ x' = ax + by + c \]  \hspace{1cm} (A2.3)
\[ y' = -bx + ay + d \]  \hspace{1cm} \text{(A2.4)}

for new transformation parameters \(a\), \(b\), \(c\) and \(d\).

To solve for the transformation parameters \(a\), \(b\), \(c\) and \(d\), at least two pairs of corresponding points in the image and the sketch map are needed:

\[
\begin{align*}
x_1' &= ax_1 + by_1 + c \\
y_1' &= -bx_1 + ay_1 + d \\
x_2' &= ax_2 + by_2 + c \\
y_2' &= -bx_2 + ay_2 + d
\end{align*}
\]

Solving for \(a\), \(b\), \(c\) and \(d\) in terms of \(x_1\), \(y_1\), \(x_2\), \(y_2\), \(x'_1\), \(y'_1\), \(x'_2\) and \(y'_2\) gives:

\[
a = \frac{(y_2 - y_1)(y_2' - y_1') + (x_2 - x_1)(x_2' - x_1')}{(y_2 - y_1)^2 + (x_2 - x_1)^2}
\]

\[
b = \frac{x_1' - x_2'}{y_1 - y_2} + a\frac{x_2 - x_1}{y_1 - y_2}
\]

\[
c = x_1' - ax_1 - by_1
\]

and

\[
d = y_1' + bx_1 - ay_1
\]

It should be noted that the transformation parameters computed by the equations in this appendix are not the ones used to register the Landsat image to the map. These parameters, computed for the best match of each pair of curves in the Landsat image and the map, are used to select a consistent subset of all the possible matches. A least squares method is then used to estimate the parameters of an affine transform using all the data gathered from the consistent matches (see section 5.1).
Appendix 2

Manual pages of a few programs in the public library

This appendix contains the manual pages for programs *curve, contours, smooth, histplot, lplot* and *peak*. These manual pages are available on-line on the LCV's Vax computer. The programs were developed as part of the work towards an M.Sc. degree and are part of the public library of programs on the same system.
NAME
curve - carry out various operations on a planar curve

SYNOPSIS
curve [option] [file-option file]

DESCRIPTION
Curve is a multi-purpose program which accepts a planar curve (consisting of one segment only) in the form of a lisp plot file (see lpf(5-UBC)) and performs one or more operations on that curve.

The input lisp plot file can be specified using the -p key. If it is not specified, it will be read from standard input. Several output files can be specified using the -o, -f, -x and -l keys as necessary. As many as 4 output files can be required for the program to run. All but one of these files must be specified using the appropriate keys. The last one, if not specified, will be written to standard output. The curve will be sampled such that each sampled point will have the same distance (equal to unity; measured on the curve) from the next one. The sampled points will be used instead of the original points throughout the program.

The following options are interpreted by curve. A value for sigma, parameter of the Gaussian function, must be specified using the -s key for all options except the -c option. When specified, sigma must be >= 1.0.

-n Smooth the curve by "convolving" it with a Gaussian function. The level of smoothness of the curve is determined by the value of sigma. The output file will be a lisp plot file and can be specified using the -o key.

-z Find zero-crossings in the curvature of the smooth curve (points of inflexion) and mark them in a file also specified by the -o key (so the zero-crossings will be marked on the smooth curve). If the -n option is not specified, only the tic marks will be there but not the curve itself.

-d Write the number of sampled points on the first line of an output file (specified using the -x key) followed by the x and y coordinates of each sampled point and the curvature of the curve at that point on each subsequent line of the file.

-k Plot the x, y and curvature functions as functions of pathlength. The output file can be specified using the -f key.

-c Find the scale-space image of the curve. The result is a binary image file which can be specified using the -l key.

AUTHOR
Farzin Mokhtarian

LIMITATIONS
Sigma should be less than 400.
There should be no more than 1024 sampled points on the curve.

The scale-space image of a very long curve can be very large and time consuming (cpu) to compute (max. 1024 x 1024).

SEE ALSO
lpf(5-UBC), image(1-UBC)
BUGS

There can be gaps at the top of contours in the scale-space image due to quantization errors. The program tries to 'fill' the gaps as much as possible but will fail if the gap is too big or if it doesn't know how to fill the gap.
NAME
contours - find the contours of black regions in a binary image

SYNOPSIS
contours [input file] [output file]

DESCRIPTION
Contours accepts a standard binary image file and finds the contours of black regions in that image. The output will be a lisp plot file (see lpf(5-UBC)). Contours is designed to also work for regions not completely contained in the image.

If no files are specified, standard input and standard output will be used. If only one file is specified, it will be assumed to be the input file and only standard output will be used.

AUTHOR
Farzin Mokhtarian

SEE ALSO
lpf(5-UBC), image(1-UBC)

LIMITATIONS
Contours uses a 4-connected definition for black regions.
NAME
smooth - smooth a function or histogram until a specified number of peaks is obtained

SYNOPSIS
smooth [-p peaks] [input-file]

DESCRIPTION
Smooth reads a function or a histogram (an image file with one row) and finds a value for sigma, parameter of a Gaussian function, such that when that function is convolved with the input histogram or function, a number of peaks specified by the -p option (defaults to 2) is obtained. The value of sigma found is actually the geometric average of the minimum and maximum values of sigma which result in the specified number of peaks. If only one peak is asked for, then the minimum value of sigma will be returned.

If the input file is not specified, standard input will be used. Value found for sigma will be written to standard output.

AUTHOR
Farzin Mokhtarian

SEE ALSO
curve(1-UBC), image(1-UBC)
NAME
histplot - change a histogram (image file with 1 row) to a lisp plot file format

SYNOPSIS
histplot [input file] [output file]

DESCRIPTION
Histplot accepts as input a histogram or a function, which must be an image file with one row, and writes it to its output as a lisp plot file (see lpf (5-UBC)). Lisp Plot Files can be plotted using lplot (1-UBC).

Standard input and standard output can be used instead of the input and output files respectively.

AUTHOR
Farzin Mokhtarian

SEE ALSO
lplot(1-UBC), histogram(1-UBC), lpf(5-UBC), image(1-UBC)
NAME
lplot - plot a lisp plot file on the imagen printer

SYNOPSIS
lplot < input-file [-s size]

DESCRIPTION
Lplot accepts a lisp plot file (see lpf (5-UBC)) from standard input and produces a plot on the imagen printer. The size of the plot can be controlled using the -s key. The number following -s must be between 0.0 and 1.0 where 1.0 specifies a full size plot (the default case) and a number between 0.0 and 1.0 specifies the size of the plot as a fraction of the full size.

AUTHOR
Farzin Mokhtarian

FILES
/usr/public/lplot Runnable file
/usr/src/public/curves/lplot.c C source file
/usr/public/iplot Runnable file
/usr/src/public/curves/iplot.c C source file
/usr/public/lib/iplot/iplt.l Lisp source file
/usr/public/lib/iplot/iplt.o Lisp object file

SEE ALSO
lpf(5-UBC)
NAME
peak - find peaks and troughs in histograms, functions

SYNOPSIS
peak [input file] [-p] [-t] [-n]

DESCRIPTION
Peak reads a histogram or a function (an image file with one row) already convolved with the first
derivative of a Gaussian function and finds the peaks (transitions from positive to negative) and
troughs (transitions from negative to positive) in that function or histogram.

If the input file is not specified, standard input will be used. By default, the number of peaks will
be written to standard output followed by the locations of those peaks. The number of troughs
and locations of the troughs will be written next.

Peak recognizes the following options:

-p Write only the number and location of peaks.
-t Write only the number and location of troughs.
-n Write the number of peaks and/or troughs only.
-m Write the locations of peaks and/or troughs only.

AUTHOR
Farzin Mokhtarian

SEE ALSO
curve(1-UBC), smooth(1-UBC), image(1-UBC)