A METHODOLOGY FOR RASTER TO VECTOR CONVERSION OF COLOUR SCANNED MAPS

OJASWA SHARMA

May 2006

TECHNICAL REPORT
NO. 240

A METHODOLOGY FOR RASTER TO VECTOR CONVERSION OF COLOUR SCANNED MAPS

Ojaswa Sharma

Department of Geodesy and Geomatics Engineering
University of New Brunswick
P.O. Box 4400
Fredericton, N.B.
Canada
E3B 5A3

May 2006

PREFACE

This technical report is a reproduction of a thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering in the Department of Geodesy and Geomatics Engineering, May 2006. The research was supervised by Dr. Darka Mioc, and support was provided by the Natural Sciences and Engineering Research Council of Canada.

As with any copyrighted material, permission to reprint or quote extensively from this report must be received from the author. The citation to this work should appear as follows:

Sharma, Ojaswa (2006). *A Methodology for Raster to Vector Conversion of Colour Scanned Maps*. M.Sc.E. thesis, Department of Geodesy and Geomatics Engineering Technical Report No. 240, University of New Brunswick, Fredericton, New Brunswick, Canada, 134 pp.

Abstract

This thesis is an attempt to automate a portion of the paper map conversion process. This includes replacing the manual digitization process by computer assisted skeletonization of scanned paper maps. In colour scanned paper maps various features on the map can be distinguished based on their colour. This research work differs from the previous research in the way that it uses the Delaunay triangulation and the Voronoi diagram to extract skeletons that are guaranteed to be topologically correct. The features thus extracted as object centrelines can be stored as vector maps in a Geographic Information System after labelling and editing. Furthermore, map updates are important in any Geographic Information System. This research work can also be used for updates from sources that are either hardcopy maps or digital raster images. The extracted features need manual editing in order to be usable in a Geographic Information System. This involves manual gap filling and clutter removal. A prototype application that is developed as part of the research has been presented. This application requires a digital image as input and processes it to produce skeletons or boundaries of objects. This research work can be further extended by considering automated gap filling in the extracted features.

Acknowledgments

It is a great pleasure to thank all those who made this thesis possible. I would first like to thank my supervisors Dr. Darka Mioc and Dr. François Anton. Apart from financially assisting me, Dr. Mioc has always been a great source of enthusiasm and support. I really appreciate her practical suggestions that always "work". She has been more than a supervisor to me. I would like to thank my co-supervisor, Dr. Anton, for the fruitful discussions that helped me understand the basics of the Voronoi digaram and the quad-edge data structure. The salad (the source code) was not easy to digest without his help. My sincere gratitude to both of them.

I would like to acknowledge my friends who have been helpful during my stay in Canada so far. Many thanks to Salman, Shyam, Valarmathy, John, Girija, Yogeshwar, Santosh, Kan and others. Thanks buddies for a wonderful time in Calgary. It would be an act of injustice to leave out my best friend Srikant who, sitting in Arizona, would discuss anything with me for hours.

Lastly, I would like to express my deepest gratitude to my parents who made the foundation of what I am today. I highly appreciate their efforts in sending me to Canada for higher studies. I dedicate my thesis to my parents.

This research titled "Data structures and algorithms for the integration of raster and vector GIS" was funded by NSERC under the Discovery grants Program. Their financial support is highly appreciated.

Table of Contents

List of Figures

Chapter 1

Introduction

Automated image feature extraction is the key to real-time map updates, automated digitization and many other tasks that involve robotic vision. In this research, we are concerned with centreline extraction from colour, scanned maps. Various map features have been considered but our primary interest is in linear features, since, after point features, they provide excellent identification objects in any analysis.

Manual digitization is very time consuming and tedious in nature. The semi-automated algorithms [Lee et al., 1998] for digitization involve tracing the object from a black and white map and then picking up the centreline of the object. Other algorithms ask for human input for decisions, such as to connect lines at junctions. Interactive systems for map digitization and query, that involve human interaction with the machine, have been designed [Quek and Petro, 1993]. A fully automated approach is still an open problem since the maps are produced for human consumption and make use of the heuristic reasoning and world knowledge of human beings [Kasturi et al., 1989].

1.1 Motivation and Problem Definition

Jähne [2004, chap. 2] states that the fundamental difference between human and machine vision can be described as "humans recognize, machines measure". The author also provides a comparison of various tasks handled by human vision and machine vision. While the human brain is exposed to contextual information and a huge knowledge base, a computational system has access only to arrays of numbers in a digital image. It is not possible to simply transfer or copy the human biological vision to machine vision systems [Jähne, 2004]. With the notion of automated digitization in mind, it is intuitive to first come up with a conceptual framework for it.

This research is primarily concerned with maps and their features. Using scanned maps instead of digital images, introduces a certain degree of simplicity because of the crisp and well-defined object boundaries in maps. On the other hand, since maps are made to be read by human beings, they have labeled text over the objects that increases the error and difficulty in extracting objects automatically from it.

A good starting point in the direction of automated feature extraction is an understanding of the space around us. Objects have coordinates, but coordinates are not always important *per se*. In daily life, we do not deal with coordinates associated with objects, rather we identify object locations in relation to other objects. This gives rise to the concept of *topology* [Kelley, 1955]. As objects are placed in relation to other objects in space, there is a sense of direction and neighbourhood associated with them. For example, we never say that a piece of wood is lying at coordinate (20 m, 20 m) in a room with origin at the corner with the door. This is a very cumbersome description of the location of an object for humans. A better description would say that a piece of wood is lying on a desk in the right corner from the door entrance. The

main point here is that relative descriptions are as powerful as geometric descriptions and can even be used to build robust identification systems.

Having said that, automated algorithms need to incorporate this sense of direction and relativeness in order to produce accurate results. This moves the attention to spatial object representations that deal with topology. Algorithms involving topological relations along with geometrical properties are the focal point of this research.

1.2 Research Objectives

The first objective of this research is to design a methodology to automate delineation of the centreline and boundaries of objects from scanned maps. Under this main objective, special consideration is given to study existing skeletonization algorithms including morphological thinning, distance transform and Voronoi diagram based skeletonization algorithms and extending the skeleton extraction algorithm by Anton et al. [2001] to colour images.

The second objective includes design of an interactive system for automated centreline extraction from colour images. Such a system should enable image based object extraction and its skeletonization. Last but not the least, refinement of the skeletonization algorithm in [Anton et al., 2001] to remove spurious branches of the skeleton is an important objective.

1.3 Methodolgy

This section will give an overview of the methodology used in this research to extract object centerlines from colour scanned maps. The input to the algorithm is a digital map scanned as a 24-bit (true colour) image. The output is the centreline of a selected

map object. The following general methodology has been adopted to process colour scanned maps:

1. Segment an image to form contiguous regions representing objects.
2. Select an object and perform sampling along its boundary.
3. Compute Delaunay and Voronoi graphs from these points.
4. Compute a skeleton from these graphs.

This general procedure used to extract skeletons from scanned maps comes from the fields of image processing and computational geometry. The following chapters discuss the methodology and involved algorithms in detail.

1.4 Thesis Organization

This thesis is organized into three basic parts. The first part is the introduction and the literature review covered by the first two chapters. The second part is the discussion of the main research covered by the third and the fourth chapters. The third part presents the results and the conclusions in last two chapters.

The first chapter gives the motivation and objectives of this research. The motivation helps the reader to understand the need for this research. The section on methodology gives an overview of the methods used in this research. Later in this chapter, a few basic terms are presented and explained and lastly applications of the research are explored.

Chapter two contains the literature review for this research and includes a study of the existing colour image processing and skeletonization algorithms. Starting with previous trends in this field, various popular skeletonization methods are presented.

Edge detection and image segmentation for both gray scale and colour images is discussed followed by an efficient colour edge detection algorithm by Comaniciu and Meer [1997]. Next, the quad-edge data structure [Guibas and Stolfi, 1985] is explained in detail. This is followed by a boundary extraction algorithm by Gold [1999] and a skeleton extraction algorithm by Anton et al. [2001].

Chapter three presents colour image segmentation and its implementation in extending the algorithm in [Anton et al., 2001] to colour images. Interactive object selection from segmented images is explained along with boundary sampling from an object patch.

Chapter four discusses the skeletonization technique used in this research. Edge Pruning methods required to remove excessive branching of the skeleton are discussed. Experimental results and analysis for the adopted pruning method are presented at the end.

Chapter five presents results of the research work. The results of object extraction from colour images and the results of skeleton and boundary extraction are shown. Experiments with scanned images and satellite imagery are presented along with a brief discussion about the quality of results and time complexity of the whole procedure.

Conclusions and future work are discussed in the last chapter. Salient points of this research are summarized and a few points in the direction of future work are suggested.

Figure 1.1 shows a flowchart of the thesis organization.

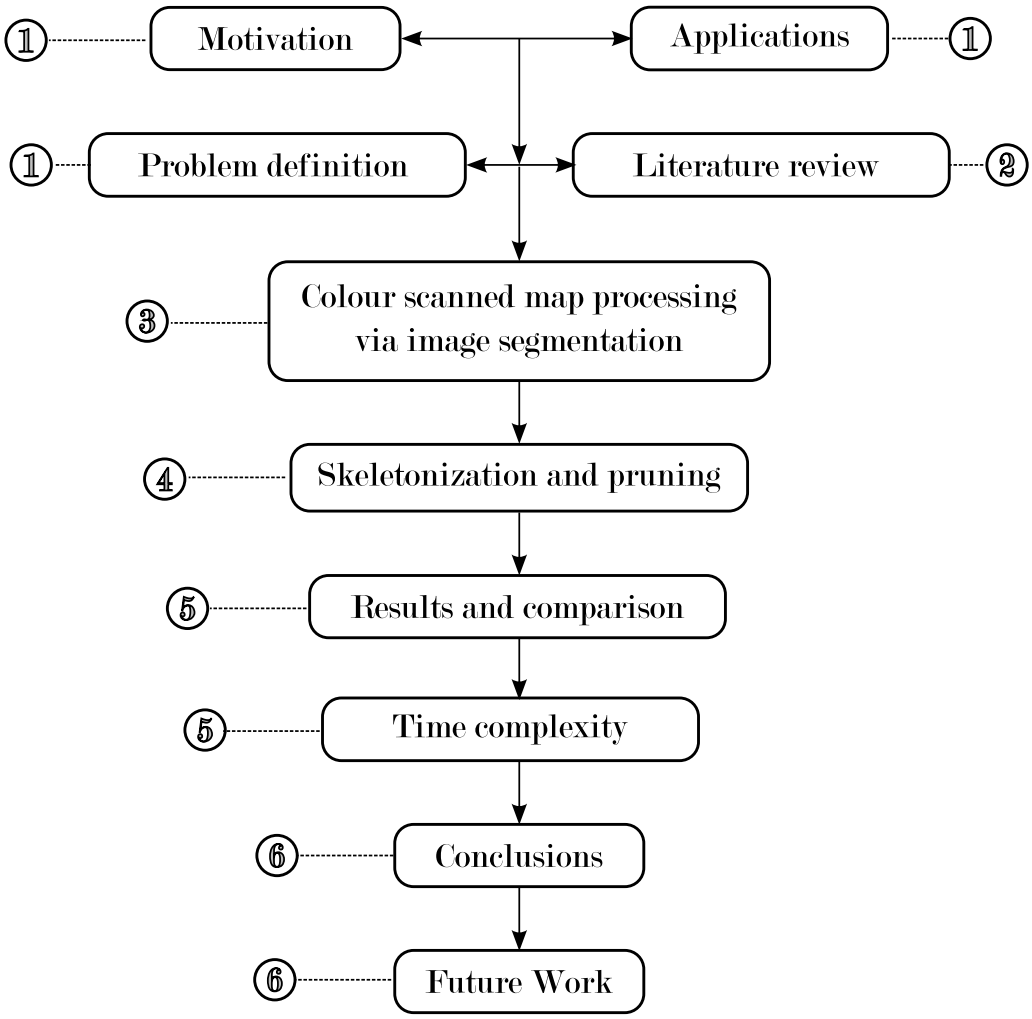


Figure 1.1: Thesis organization.

1.5 Background

This section is a brief discussion of a few terms in image processing and computational geometry that will be commonly used in this text. Basic concepts are discussed and illustrated with the help of figures.

1.5.1 Raster Image

Image representation on a computer involves a transformation from continuous space to discrete space. An image which is continuous in both spatial and spectral (colour intensity) domains is converted into a digital image by means of *sampling* and *quantization* [Gonzalez and Woods, 2002]. Sampling converts continuous coordinates into discrete coordinates (see Figure 1.2) while quantization results in discrete colour values (see Figure 1.3). The colour values are also called *Digital Numbers* (DN values).

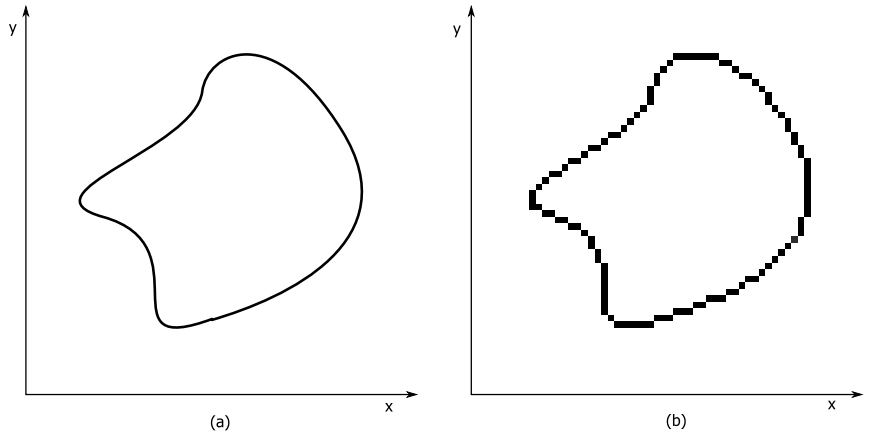


Figure 1.2: (a) Continuous curve. (b)Result of image sampling.

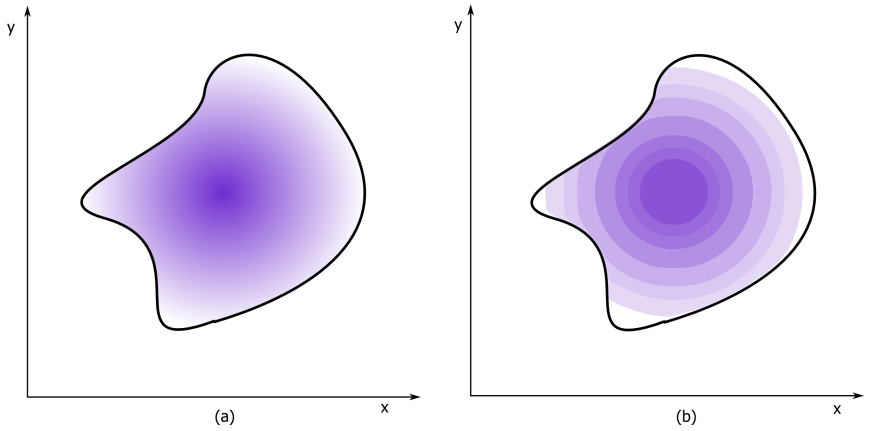


Figure 1.3: (a) Continuous tone of colours inside a curve. (b) Result of quantization.

Thus, a raster image is an array of numbers representing colour values (see Figure 1.4). Each element of this array is called a *Pixel* (*Pi*cture *El*ement). In matrix notation, an $M \times N$ image I is represented as [Gonzalez and Woods, 2002]:

$$I = \begin{bmatrix} i_{0,0} & i_{0,1} & \cdots & i_{0,N-1} \\ i_{1,0} & i_{1,1} & \cdots & i_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ i_{M-1,0} & i_{M-1,1} & \cdots & i_{M-1,N-1} \end{bmatrix} \tag{1.1}$$

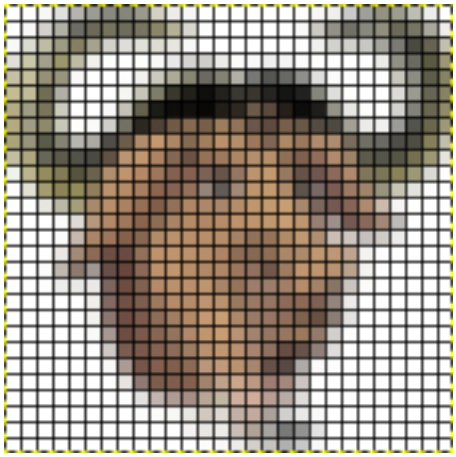


Figure 1.4: Raster image showing individual pixels.

The number of discrete levels of colour L in an image is generally an integer power of 2

$$L = 2^k \tag{1.2}$$

where, k is the number of bits used to store a colour value. This distinguishes three important classes of digital images based on their storage [Gonzalez and Woods, 2002, chap. 2].

Monochrome image is an image with only 2 levels ($k = 0$) of colour, namely black and white. It is also known as a **binary image**(see Figure 1.5(a)).

Gray scale image is an image with a total of 256 possible colours ($k = 8$). Generally these colours represent shades of gray varying from black (DN value of 0) to white (DN value of 255). Nevertheless, such images can be indexed to store colour images with the DN values mapped to a particular display colour via a colour palette table. Figure 1.5(b) shows a gray scale image.

True colour image is an image with 16 million colours ($k = 24$). In practice, such images are coloured and store colour values as 8-bit triplets representing the red, green and blue components (see Figure 1.5(c)).

(a) Binary image (b) Gray scale image (c) True colour image

Figure 1.5: Colour depth in digital images.

1.5.2 Vector Image

A Vector image is a representation of the real world in the form of transformable objects. These objects are composed of primitive shapes like points, lines and shapes with editable attributes like colour, fill and outline. Objects in vector model are defined by mathematical equations rather than pixels. Since objects can be freely transformed (scaled, rotated, etc) without any loss of detail, vector images are resolution (or scale) independent (see Figure 1.6).

(a) Original image

(b) Scaling by a factor of 0.35

(c) Rotation by 30 degrees clockwise

Figure 1.6: Scale independence in vector images.

1.5.3 Map Feature

A map feature is a homogeneous area or *patch* on a map representing a ground feature, phenomenon or object. Thus a map feature can be a road, river or field. Figure 1.7 shows a few map features.

1.5.4 Skeleton

A skeleton of an object can be described as its centreline. It is one of the important topological descriptors of an object. In simple words, skeletons provide a representation of a shape that is as thin as possible and that lies near the middle of many portions of a shape [Costa and Cesar Jr., 2001, chap. 5].

A more formal definition of skeleton is based on the concept of maximal inscribed disk [Sonka et al., 1999, chap. 11]. A **disc** $D(p, r)$ with center p and radius r, $r \geq 0$, is a set of points with distances d from the center less than or equal to r. An **inscribed** disc is a disc that lies completely within the shape under consideration. The disc D from a set S of inscribed discs is said to be **maximal** if and only if there is no larger

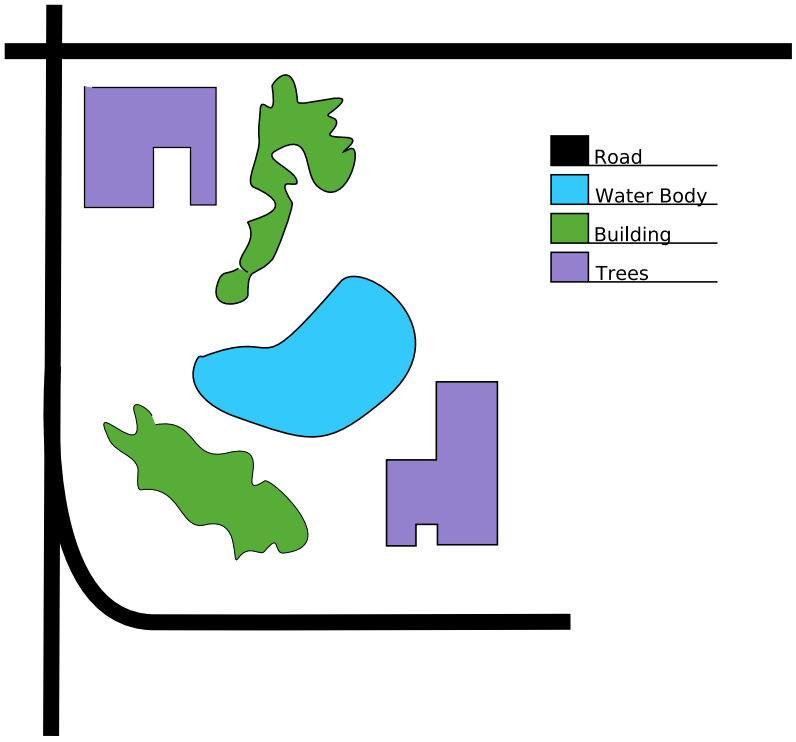


Figure 1.7: Map features.

disc that contains D. As shown in Figure 1.8, discs $D1$, $D2$, and $D4$ are maximal while $D3$ is not.

Skeleton by maximal discs is the locus of the centre of maximal inscribed discs (see Figure 1.8). Thus the skeleton S can be written as [Sonka et al., 1999, chap. 11]:

$$S(X) = \{p \in X : \exists r \geq 0, D(p, r) \text{ is a maximal disc of } X\} \tag{1.3}$$

Sonka et al. [1999, chap. 11] discuss two drawbacks of skeleton by this definition. The resulting skeleton does not necessarily preserve connectivity of the original object set and some of the skeleton lines may be wider than one pixel in the discrete plane.

A more refined definition by Blum [1967] explains skeleton by means of *medial axis transform* (MAT). The medial axis is often achieved using the distance transform

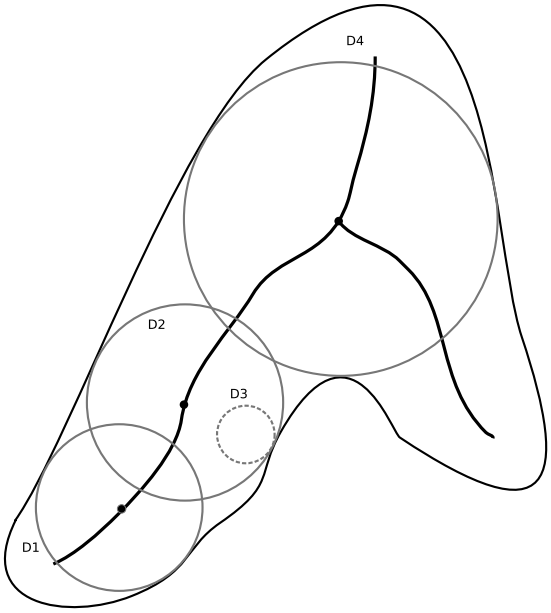
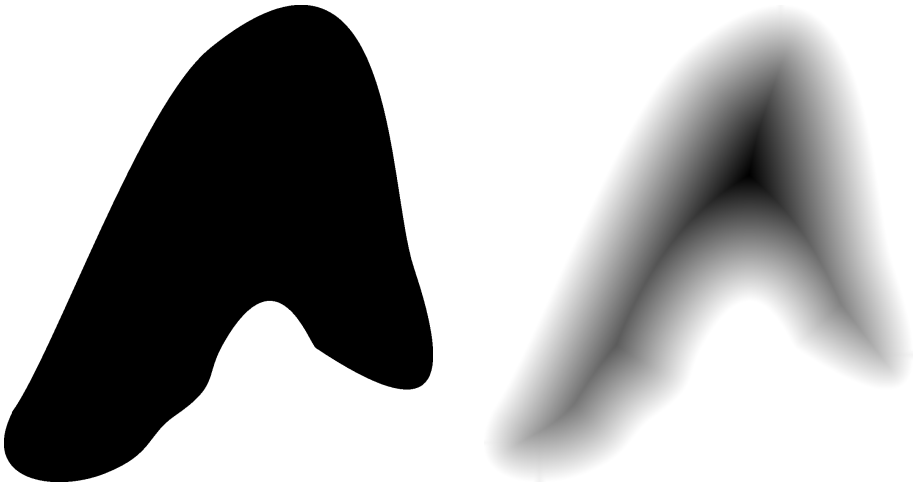


Figure 1.8: Shape skeleton showing maximal inscribed discs.

as shown in Figure 1.9.



(a) Object (b) Distance transform

Figure 1.9: Skeleton via distance transform.

1.5.5 Object Sampling

In context of this research, object sampling essentially means sampling the boundary [Gold, 1999] of an object patch. This is done by collecting points along the boundary of an object (see Figure 1.10).

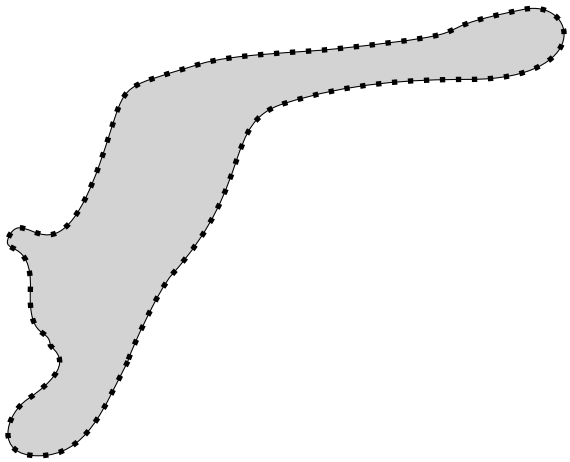


Figure 1.10: Boundary sampling.

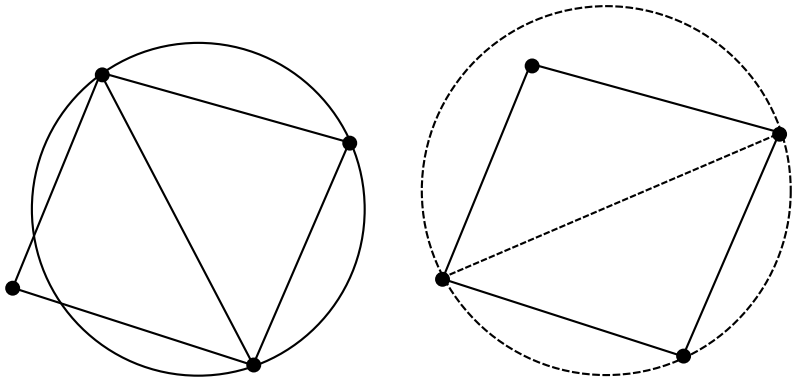
A dense sampling ensures better results by representing the object accurately. In digital space, points in a set of edge detected boundary of the object provide sufficiently well sampled points.

1.5.6 Delaunay Triangulation

The *Delaunay triangulation* [O'Rourke, 1998; de Berg et al., 2000] $\mathscr{D}$ of a set of points P in $\mathbb{R}^2$ is a *triangulation* such that no point in P is inside the circumcircle of any triangle of the triangulation. It was first invented by the Russian number theorist, Boris Nikolaevitch Delone [Delaunay, 1934].

A triangulation is a subdivision of a geometric object into triangles. In 2-dimensional Delaunay triangulation, the *empty circle* criterion (see Figure 1.11) maximizes the minimum angles of all the triangles in the triangulation [Okabe et al.,

1992, page 112].



(a) Delaunay triangulation (b) Not a Delaunay triangulation

Figure 1.11: Empty circle criterion.

A few methods for computing the Delaunay triangulation include the flipping algorithm [Sibson, 1977], divide-and-conquer algorithm [Dwyer, 1987], plane sweep algorithm [Fortune, 1987] and randomized incremental insertion algorithm [Guibas et al., 1992]. A good comparison of algorithms to compute Delaunay triangulations is provided by Krämer [1995]. An intuitive relationship between Delaunay triangulations in the plane and the convex hull of a particular set of points in 3D space was shown by Brown [1979]. This relationship shows that it is possible to compute the Delaunay triangulation and the Voronoi diagram of a set of points in the plane from the convex hull of a set of points in space. Given a set S of n points in the plane $z = 0$, let S' denote a set of points in 3D space such that each point in S is projected on a unit elliptic paraboloid $z = x^2 + y^2$. Therefore, each point $p = (p_x, p_y) \in S$ is projected to a point $p' \in S'$ as:

$$p' = (p_x, p_y, p_x^2 + p_y^2) \tag{1.4}$$

Back projecting the edges of the downward facing convex hull of points in S' will

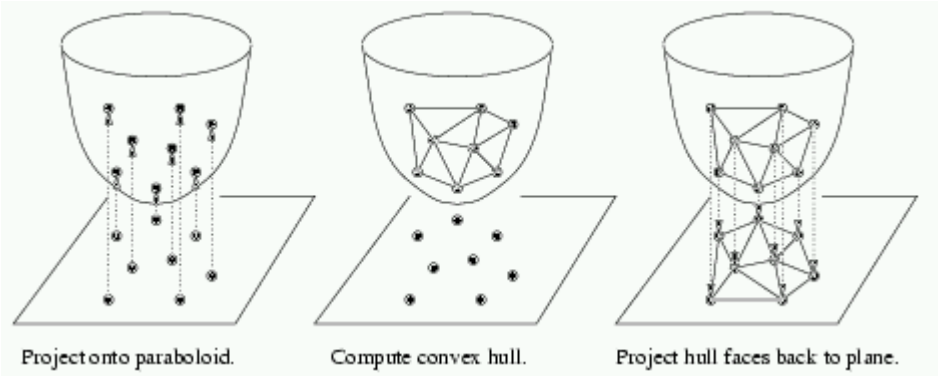


Figure 1.12: Delaunay triangulation from convex hull (from: Pless [2003]).

give the Delaunay triangulation of points in S. This is illustrated in Figure 1.12.

In order to formulate a criteria to test if a triangle satisfies the condition for Delaunay triangulation, consider four points p_a, p_b, p_c, and p_d in a plane. Point p_d is outside the circle passing through p_a, p_b, p_c if the predicate $H(p_a, p_b, p_c, p_d) \geq 0$ (see equation 1.5). Therefore, this becomes the basic formulation of the incircle test in the computation of Delaunay triangulation (see Figure 1.13).

$$H(p_a, p_b, p_c, p_d) = \begin{vmatrix} 1 & x_a & y_a & x_a^2 + y_a^2 \\ 1 & x_b & y_b & x_b^2 + y_b^2 \\ 1 & x_c & y_c & x_c^2 + y_c^2 \\ 1 & x_d & y_d & x_d^2 + y_d^2 \end{vmatrix} \tag{1.5}$$

1.5.7 Voronoi Diagram

A Voronoi diagram of a set of points P in a plane is an irregular partitioning of space into polygonal regions. Each region belongs to a generator point in P (see Figure 1.14). Any point inside a Voronoi region R_i is closer to the generator $p_i \in P$ of R_i than to an any other point in P.

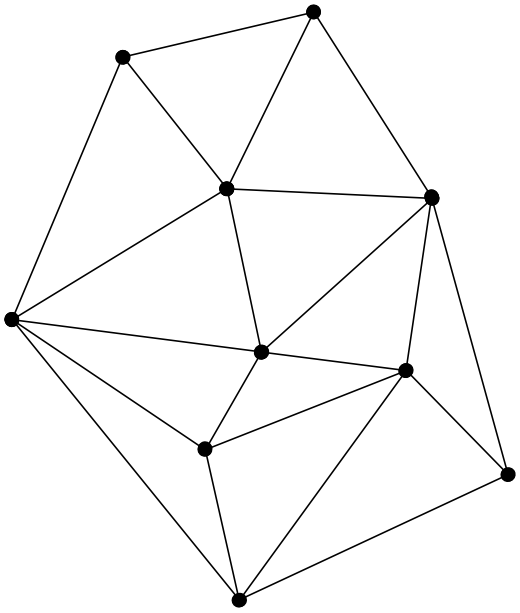


Figure 1.13: Delaunay triangulation.

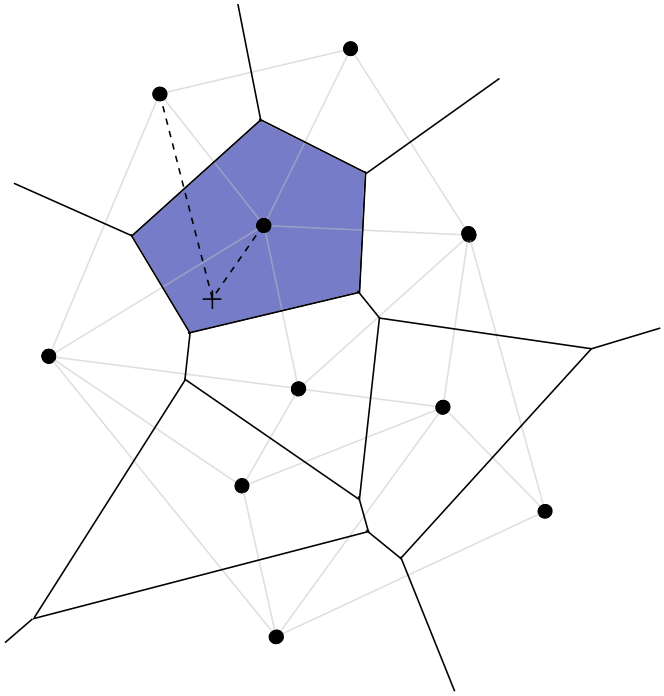


Figure 1.14: Voronoi Diagram.

Okabe et al. [1992] discuss the Voronoi diagram and its properties in detail. In mathematical terms [Okabe et al., 1992, chap. 2]: Let $P = \{p_1, \ldots, p_n\}$ be a set of distinct points, where $2 \leq n < \infty$ and $x_i \neq x_j$ for $i \neq j$, $i, j \in I_n$. Here I_n is the set of natural numbers from 1 to n. The region given by

$$V(p_i) = \{x | \, \|x - x_i\| \leq \|x - x_j\| \text{ for } j \neq i, j \in I_n\} \tag{1.6}$$

is called the *Voronoi polygon* associated with the *generator* p_i, and the set given by

$$\mathscr{V} = \{V(p_1), \ldots, V(p_n)\} \tag{1.7}$$

is called the *Voronoi diagram* generated by P.

1.6 Applications of the Proposed Research

The proposed research has a wide range of applications. Apart from the field of GIS, this research holds importance in the field of Photogrammetry and Computer Science. The key areas of application are:

- Automated map feature digitization in a GIS environment
- Map updates in GIS and change detection
- Extraction of natural linear features from satellite imagery
- Robotic vision

The primary application lies in the area of automated digitization. Automated digitization is in demand since digital maps need to be updated frequently due to

changes like urbanization, residential developments, constructions and other phenomena. Manual digitization is very strenuous and time consuming process [Bin and Cheong, 1998]. It is therefore required to automate the process of digitization.

The second application is in the field of GIS map updates. An automated feature extraction system can quickly extract features from updated maps. These features can then be used for updating the existing maps in a GIS. Change analysis can also be performed.

The latest research in photogrammetry [Habib et al., 2003] attempts to perform automated image matching in a stereo-pair. This needs good intersection points and linear features provide excellent intersections. Therefore, the process of extracting linear features and automatically obtaining their centrelines is gaining importance. This research can extract simple natural features from satellite imagery like coastlines. These can be used in photogrammetry.

Robotic vision [Vlassis et al., 2000] is a research area in Computer Science. This requires objects to be identified by using digital cameras. Centerlines are one of the prime object descriptors. Once the object image has been captured by a camera, it is then sent to an identification system that tries to obtain information about the object. Automated centreline extraction can help with object identification.

Chapter 2

Previous Research

This chapter explores the work that has already been done in the field of feature extraction. Special consideration has been given to centreline extraction methods that are based on the Delaunay triangulation and Voronoi diagram. Basic data structures and image processing operations are reviewed here. Starting with the previous research on feature extraction, the discussion drifts towards the methods of describing object boundaries in an image. This builds the foundation for the later part where existing object skeletonization methods are described.

2.1 Previous Research on Feature Extraction

Feature extraction is an operation to extract various image features for identifying or interpreting meaningful objects from images. A more general definition of Feature extraction is [Bock, 1998]:

"The transformation of signal or image raw data into higher-level characteristic variables. These may be general features, which are evaluated to ease further processing, or application-oriented, like those needed for image recognition."

This research is concerned with extracting skeletons of objects. As discussed

by Bernard and Manzanera [1999], the skeletons produced by any skeletonization algorithm are expected to have the following properties.

Homotopy

The skeletons must preserve topology of the original object [Kong and Rosenfeld, 1989] resulting in same number of

- connected components (4-connectivity or 8-connectivity)
- holes and cavities

One-pixel thickness

The skeleton should be composed of one pixel thick curved segments.

Mediality

The skeleton should be centered within the object with every point of the skeleton the same distance from two closest borders of the object.

Rotation invariance

The skeleton of a transformed object should be the same as the transformed skeleton of the original object. In a discrete plane, this property is hard to enforce but this can still be satisfied approximately.

Noise immunity

The skeleton should have weak sensitivity to noise in the image. This also extends to insensitivity of the skeleton to addition or deletion of boundary pixels.

Reconstructibility

This refers to the ability to recover the original object from the skeleton.

In practice, however, all of these properties cannot be satisfied simultaneously [Bernard and Manzanera, 1999]. Some of the properties are even contradictory, like

- **Homotopy versus one pixel thickness**: Homotopy may result in arbitrary thickness of the skeleton in order to preserve connectivity and cavities which violates the one pixel thickness criterion.

- **Mediality versus one pixel thickness**: Mediality is an obstacle to one pixel thickness in the case when the object has a width of an even number of pixels (a pathological case is an object two pixels wide).

Therefore, the choice of properties is entirely dependent on the application. Discussed next are a few raster algorithms for skeletonization.

2.1.1 Skeletonization using thinning

Mathematical morphology is a tool for extracting image components that are useful in the representation and description of region shape, such as boundaries, skeletons, and the convex hull [Gonzalez and Woods, 2002, chap. 9]. Binary images are images that have just two colours. An object in a binary image represents a set in mathematical morphology. These sets are members of the 2-D integer space Z^2, where each member of a set is a 2D vector whose components are the (x, y) coordinates of an object pixel in the image. Logical operations on binary images provide a powerful means of image processing algorithms known as *Morphological operations* [Serra, 1982]. The principal logic operations used in image processing are AND, OR, and NOT (complement). These operations can be combined to form any other logical operation. There are two

operations fundamental to morphological processing: *dilation* and *erosion* [Gonzalez and Woods, 2002, chap. 9].

Dilation

The morphological transformation **dilation** $X \oplus B$ is the point set of all possible vector additions of pairs of elements, one from each of the sets X and B [Sonka et al., 1999, chap. 11]. The set B is called a *Structuring Element* which is a small binary image (a set in mathematical morphology) as shown in Figure 2.1(b).

$$X \oplus B = \left\{ p \in Z^2 : p = x + b, x \in X \text{ and } b \in B \right\} \tag{2.1}$$

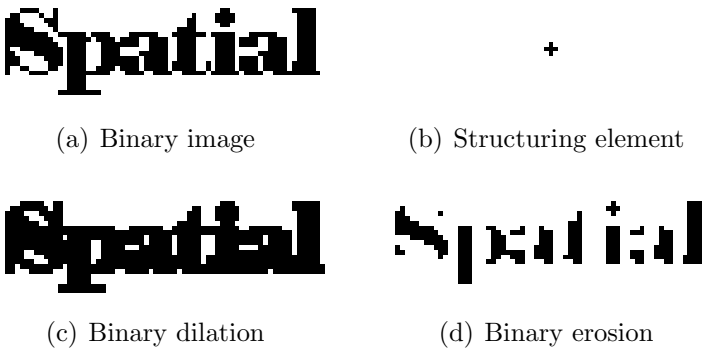
The result of dilating a binary image X by a structuring element B is the addition of a layer of pixels around the binary object in X. Thus, it causes an increase in size of the object depending on the structure of B (see Figure 2.1(c)).

Erosion

The morphological transformation **erosion** $X \ominus B$ combines two sets X and B using vector subtraction [Sonka et al., 1999, chap. 11]. Erosion is the dual operator of dilation.

$$X \ominus B = \left\{ p \in Z^2 : p + b \in X \text{ for every } b \in B \right\} \tag{2.2}$$

An eroded image is given by those points p for which all possible $(p + b)$ are in X. Thus, it causes a shrinkage in the size of the object depending on the structuring element B (see Figure 2.1(d)).



(a) Binary image (b) Structuring element

(c) Binary dilation (d) Binary erosion

Figure 2.1: Binary dilation and erosion.

Opening and closing

Erosion and dilation are not inverse transformations of each other, rather they produce simplified versions of the original image. Two important transformations called *opening* and *closing* [Gonzalez and Woods, 2002, chap. 9] are worth mentioning here. Opening effectively smooths an object by removing small protrusions while closing smooths the object by filling gaps and holes.

Erosion followed by dilation results in **opening**. The opening of an image X by the structuring element B is denoted by $X \circ B$ is defined as

$$X \circ B = (X \ominus B) \oplus B \tag{2.3}$$

Similarly, *closing* of an image X by the structuring element B, denoted by $X \bullet B$, is defined as

$$X \bullet B = (X \oplus B) \ominus B \tag{2.4}$$

Thinning is an operation that removes the foreground pixels of a binary object so that the resulting object is a one pixel thick centreline representation of the object (see Figure 2.2). Thinning is generally performed by means of morphological

operations. This is accomplished by a series of erosions and openings [Serra, 1982].

$$S(X) = \bigcup_{k=0}^{K} S_k(X) \tag{2.5}$$

with

$$S_k(X) = (X \ominus kB) - (X \ominus kB) \circ B \tag{2.6}$$

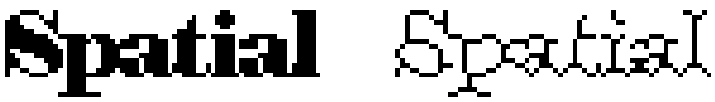
In equation 2.6, $(X \ominus kB)$ denotes k successive erosions of X as defined below.

$$X \ominus kB = \underbrace{(\cdots (X \ominus B) \ominus B) \ominus \cdots) \ominus B}_{k \text{ times}} \tag{2.7}$$

K in equation 2.5 is the last iterative step before I erodes to an empty set, given by

$$K = max\,\{k | (X \ominus kB) \neq \varnothing\} \tag{2.8}$$

This method preserves the topology [Zhou and Toga, 1999] of the object but the resulting skeleton is not guaranteed to be a true skeleton in the maximal disc sense and requires intricate heuristics to maintain skeleton connectivity [Telea and van Wijk, 2002].

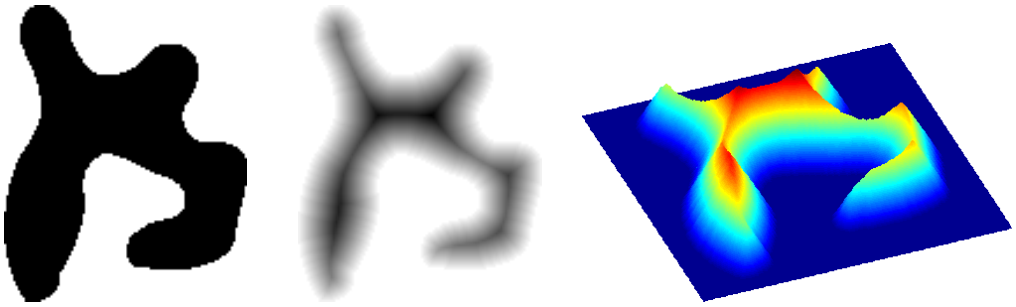


(a) Binary image (b) Thinned image

Figure 2.2: Morphological thinning

2.1.2 Skeletonization using Distance Transform

This method computes a distance transform of the boundary of the object [Borgefors, 1984]. A distance transform of the boundary of a binary object is a gray scale image that resembles the original object but the interior of the foreground pixels have a value proportional to the distance from its nearest boundary. This leads to the highest value in the middle of the object. Therefore, the skeleton lies along the singularities (i.e. creases or curvature discontinuities) in the distance transform (see Figure 2.3). This type of skeleton is related to the grass fire analogy [Blum, 1967]. A skeleton obtained via the medial axis transform will always be in the middle of the object, but will not necessarily guarantee connectivity [Reinders et al., 2000]. Furthermore, direct computation of singularities is numerically unstable and delicate [Niblack et al., 1992; Siddiqi et al., 1999].



(a) Binary image (b) Distance transform (c) Distance transform rendered as a surface

Figure 2.3: Ridges in distance transform.

2.1.3 Skeletonization using Delaunay/Voronoi graphs

Delaunay triangulation and Voronoi diagram based methods fall under the category of *Geometric methods* producing an accurate connected skeleton [Telea and van Wijk,

2002]. The Voronoi diagram of a set of points sampled along the boundary of the object contains the skeleton of the object [Ogniewicz, 1995; Ogniewicz and Kübler, 1995; Amenta et al., 1998; Gold, 1999]. Algorithms in this category try to extract the skeleton by removing spurious edges from the Voronoi diagram [Gold, 1999] or label them using some criteria [Ogniewicz, 1995]. A Voronoi diagram based approach for skeletonization is the main focus of this research and is discussed in more detail later in this chapter.

The next section presents edge detection techniques in the context of this research.

2.2 Edge Detection

In order to extract skeletons from an image, the first step is to obtain object boundaries. The implementation of skeletonization by Anton et al. [2001] uses *edge detection* to define object boundaries and perform object sampling. This research initially used the same approach for object sampling and extended the concept to colour images. Efficient methods of colour image edge detection were studied during the course of this research. Later in this research, a better alternative for object definition was sought while still using edge detection to get sample points around the object.

Edge Detection is the process of locating edge pixels in an image. In other words, edge detection gives meaningful discontinuities in gray level [Gonzalez and Woods, 2002, chap. 10]. The basis of edge detection is differentiation. Using the concept of derivatives, a two dimensional, continuous image intensity function $I(x, y)$ can be differentiated to obtain the gradient $\nabla I(x, y)$ [Gonzalez and Woods, 2002, chap. 3] as

$$\nabla I(x, y) = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} \tag{2.9}$$

At times it is important just to measure the magnitude of this change. The differential operator called the *Laplacian* may be used in that case. The Laplacian is the simplest *isotropic* derivative operator [Rosenfeld and Kak, 1982] which means that it has same properties in all directions that makes it invariant to rotation in the image. It is defined as

$$\nabla^2 I(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{2.10}$$

The differential operators discussed above are applicable to continuous functions, while our digital images are discrete in nature. Therefore, equations 2.9 and 2.10 must be approximated by *differences*. The following expressions show the discrete form of derivative operators [Gonzalez and Woods, 2002, chap. 3]. Here, $I(x,y)$ refers to the intensity value at pixel location (x,y).

$$\Delta I(x,y) = \begin{bmatrix} I(x,y) - I(x-1,y) \\ I(x,y) - I(x,y-1) \end{bmatrix} \tag{2.11}$$

$$\Delta^2 I(x,y) = \begin{pmatrix} I(x+1,y) + I(x-1,y) - 2I(x,y) \\ + \\ I(x,y+1) + I(x,y-1) - 2I(x,y) \end{pmatrix} \tag{2.12}$$

An overview of edge detection algorithms for the three types of digital images discussed in subsection 1.5.1 is presented next.

2.2.1 Binary Image Edge Detection

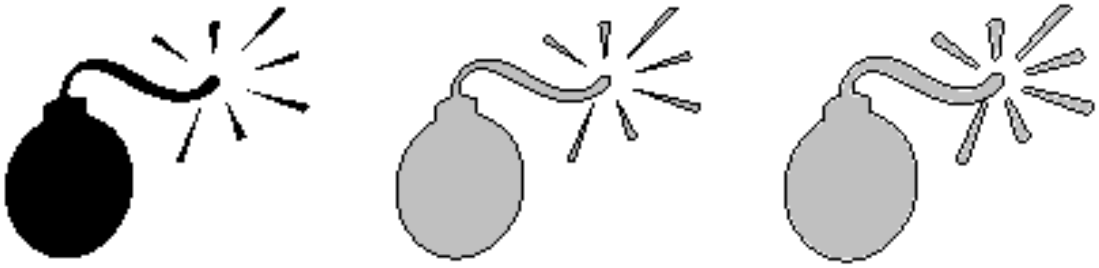
An edge in a binary image is composed of those pixels that have at least one neighbour that belongs to the background [Parker, 1997, chap. 2]. Using dilation and erosion, there are two possibilities of delineating edges:

- **Inner edges**: These edges stick to the boundary of the object from inside. Inside edges E_{in} can be extracted by subtracting an eroded image from the original one (see Figure 2.4(b)). The structuring element B used in the examples shown in Figures 2.4, 2.5 and 2.6 is a 3×3 diamond shaped matrix as shown in Figure 2.1(b). Inner edges are suitable for areal objects since they result in boundaries that are well separated from those of other areal objects as shown in Figure 2.5.

$$E_{in} = I - (I \ominus B) \tag{2.13}$$

- **Outer edges**: These edges wrap the object from the outside. The original image subtracted from the dilated image will result in the outer boundary E_{out} (see Figure 2.4(c)). Outer boundaries are suitable for thin linear features since the resulting edges are guaranteed to have one pixel space between the boundary as shown in Figure 2.6. This is necessary to obtain the centreline in an extreme case where the object itself is one pixel wide.

$$E_{out} = (I \oplus B) - I \tag{2.14}$$



(a) Binary image (b) Inner edges (c) Outer edges

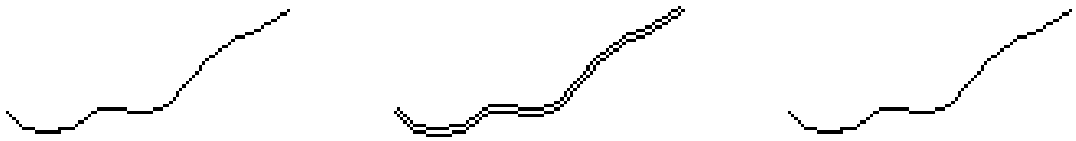
Figure 2.4: Binary edge detection using morphological operations.

(a) Areal objects

(b) Inner edges resulting in well separated boundaries

(c) Outer edges resulting in fused boundaries

Figure 2.5: Suitability of inner edges for areal objects.



(a) one pixel thick linear object

(b) Outer edge resulting in well spaced boundary

(c) Inner edge resulting in collapsed boundary

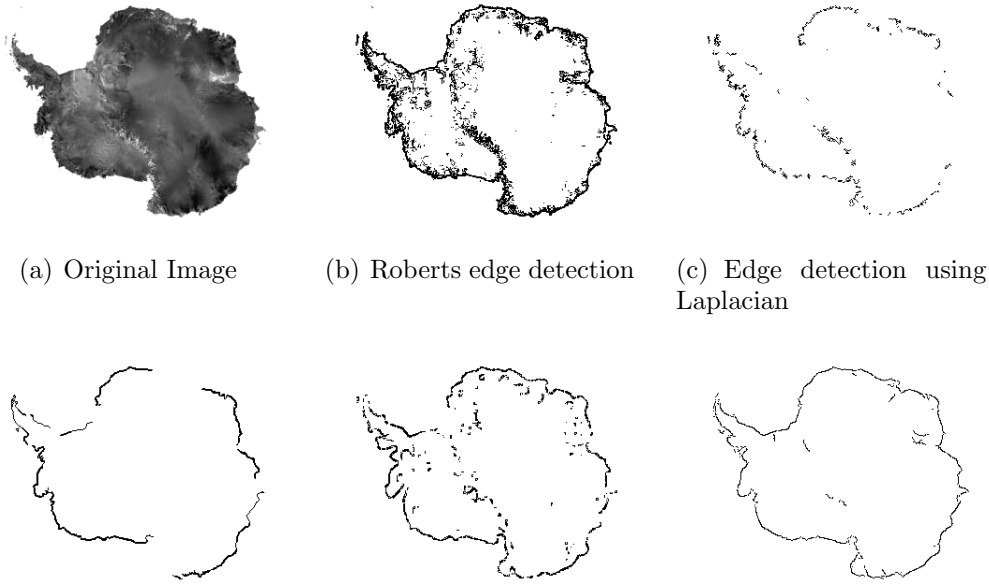
Figure 2.6: Suitability of outer edges for linear objects.

2.2.2 Gray Scale Image Edge Detection

Edge detection in the gray scale domain is different from that in the binary domain. The presence of more colours makes it possible to represent rich textural variations and other object details in the image. Edge detection in the spatial domain is generally performed by *linear filtering* which is similar to *convolution* [Jähne, 2004] in the frequency domain. Linear filtering of an image I with a filter mask w of size $m \times n$, where m and n are odd integers, is given by the expression [Gonzalez and Woods, 2002, chap. 3]:

$$I'(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) I(x+s, y+t) \tag{2.15}$$

where, $a = (m-1)/2$ and $b = (n-1)/2$. The filter mask w is also known as *convolution kernel* or *filter mask*. Filtering is accomplished by considering a neighbourhood around a pixel in an image and adding the neighboring values to the central value after multiplying them by the corresponding weights in the filter mask. Equation 2.15 has to be applied to all the pixels $I(x, y)$ of the image for $x = 0, 1, \ldots, M-1$ and $y = 0, 1, \ldots, N-1$. A wide range of spatial filtering operations have been defined this way [Gonzalez and Woods, 2002]. These include image smoothing, sharpening and edge detection. The respective results of edge detection using the Roberts, Laplace, Sobel, Marr-Hildreth and Canny, produced by the application developed in this research, are shown in Figure 2.7.



(a) Original Image (b) Roberts edge detection (c) Edge detection using Laplacian

(d) Sobel edge detection (e) Marr-Hildreth edge detection (f) Canny edge detection

Figure 2.7: Gray scale edge detection by various methods.

2.2.3 Colour Image Edge Detection

Colour is a powerful descriptor of objects in a scene. It is far easier to discern fine object details in a colour image than it is to do the same in a gray scale image [Gonzalez and Woods, 2002, chap. 6]. Figure 2.8 shows the range of visible light in the electromagnetic spectrum. Digital colour images are represented by means of various *colour models*, the most famous being the RGB model. The purpose of a colour model is to facilitate the specification of colours in a standard way.

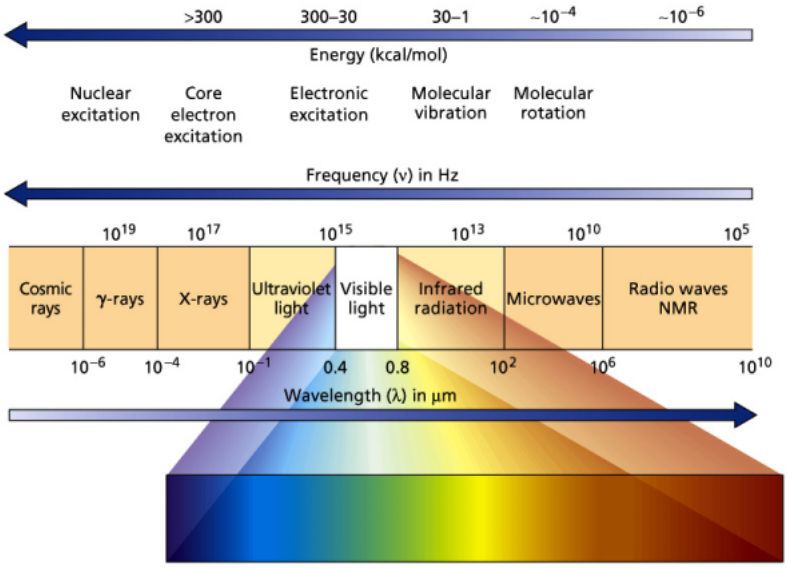
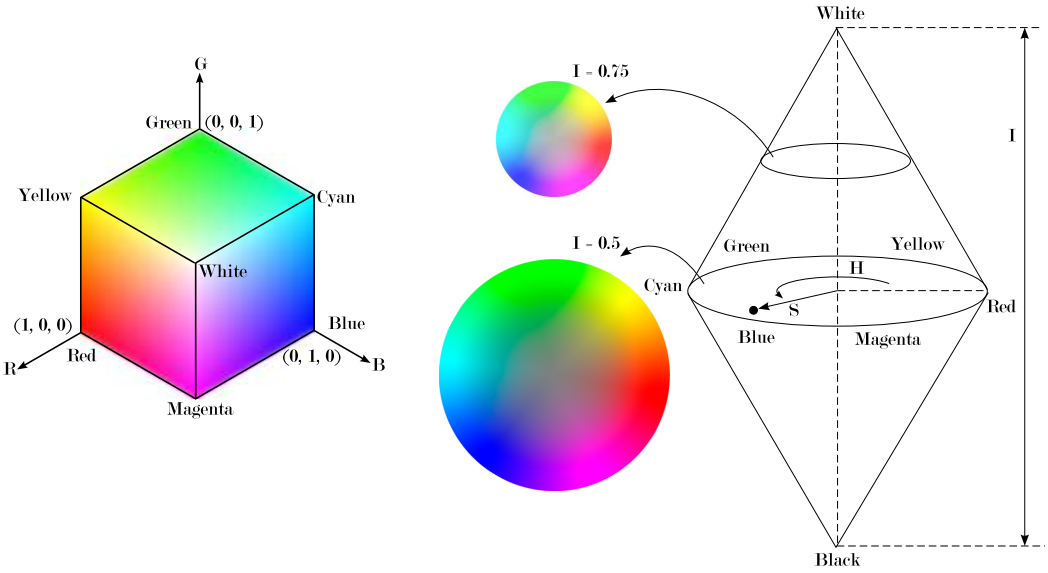


Figure 2.8: The electromagnetic spectrum. (from: Huskey [2002])

The RGB colour model

In RGB model, each colour appears in its primary spectral components of red, green, and blue [Gonzalez and Woods, 2002, chap. 6]. All the possible colours can be represented in a colour cube as shown in Figure 2.9(a). This model is based on a Cartesian coordinate system.



(a) The RGB colour cube (b) The HSI colour cone

Figure 2.9: The RGB and HSI colour models.

The CMY colour model

Cyan, magenta and yellow are secondary colours of light and primary colours of pigments. A mixture of these three colours in equal proportions will produce the black colour. Most devices that deposit coloured pigments on paper, such as colour printers and copiers, require data in a CMY colour model or convert the RGB data to CMY. This conversion is performed as [Gonzalez and Woods, 2002, chap. 6]

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.16}$$

The HSI colour model

The way human beings perceive and define colours is different than the RGB or CMY representations. It is easier to describe a colour by its hue, saturation and intensity.

The HSI (hue, saturation and intensity) colour model [Gonzalez and Woods, 2002, chap. 6] is based on a similar idea. The hue of a colour describes a pure colour, whereas the saturation gives a measure of the degree to which the pure colour is diluted by white light. Intensity gives a measure of the amount of light that distinguishes between a dark colour and a light one. Figure 2.9(b) shows the HSI model based on circles.

Given a RGB colour value, the H component can be obtained as [Gonzalez and Woods, 2002, chap. 6]

$$H = \begin{cases} \theta & \text{if } B \leq G \\ 360 - \theta & \text{if } B > G \end{cases} \tag{2.17}$$

where,

$$\theta = \cos^{-1}\left\{ \frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-B)(G-B)\right]^{1/2}} \right\}$$

The saturation component is given by

$$S = 1 - \frac{3}{(R+G+B)}\left[\min\left(R,G,B\right)\right] \tag{2.18}$$

and the intensity component is given by

$$I = \frac{1}{3}(R+G+B) \tag{2.19}$$

This background in colour models will help in understanding the colour edge detection techniques presented next and the segmentation algorithm presented after that. Three methods of colour edge detection explored during the course of this research are discussed next.

Merging RGB edge detection components

This method [Gonzalez and Woods, 2002, chap. 6] is a simple extension of the gray scale edge detection methods to colour images. Here, individual red, green, and blue planes of a digital colour image are treated as gray scale images to detect the edges. The three edge images are then merged logically (using AND or OR operation) to produce the final edge image. This is shown conceptually in Figure 2.10. Results of edge detection by this method are shown in Figure 2.11(b, c, and d).

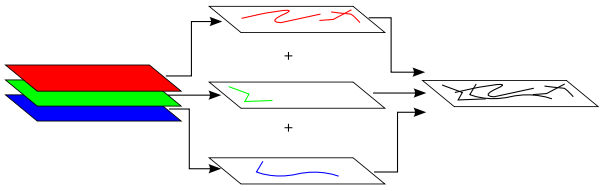


Figure 2.10: Colour Edge detection by merging edges from individual colour planes.

Vector angle modified Robert's edge detector

Dony and Wesolkowski [1999] introduce a new approach to edge detection in colour images by using the vector angle between the colours of two adjacent pixels. The vector angle enables chromaticity differentiation in a colour image. As discussed earlier, chromaticity or hue is the main descriptor of object boundaries in a colour image and hence it must be considered in the edge detection process.

Given two colour vectors $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ in the RGB colour space, Dony and Wesolkowski [1999] propose to calculate the sine of the vector angle between them as

$$\sin\theta = \left(1 - \frac{\overrightarrow{v_1} \cdot \overrightarrow{v_2}}{\|\overrightarrow{v_1}\| \, \|\overrightarrow{v_2}\|}\right)^{1/2} \tag{2.20}$$

The authors suggest the use of sine of the angle over angle θ because of computation of inverse cosine involved in it. Sine of the angle is preferred over cosine of the angle

since the sine increases with increase in the angle (up to the orthogonal angle of 90°). The Roberts operator is then extended for edge detection in colour images. It can be calculated using the maximum sine of the angles between diagonally adjacent pixels in a 2×2 block as

$$S_R^2 = 1 - \max\left(\frac{\overrightarrow{v}(x,y) \cdot \overrightarrow{v}(x+1,y+1)}{\| \overrightarrow{v}(x,y) \| \, \| \overrightarrow{v}(x+1,y+1) \|}, \frac{\overrightarrow{v}(x,y+1) \cdot \overrightarrow{v}(x+1,y)}{\| \overrightarrow{v}(x,y+1) \| \, \| \overrightarrow{v}(x+1,y) \|} \right) \quad (2.21)$$

Figure 2.11(e) shows the result of this method.

Vector angle modified Laplacian edge detector

Bakker and Schmidt [2002] provide a variant of the Laplace edge detection operator based on vector angle between colours in a 3×3 neighbourhood. The definition of the vector angle between vectors $\overrightarrow{v}$ and $\overrightarrow{w}$ refereed to as the *spectral angle* is same as defined in equation 2.20 represented as

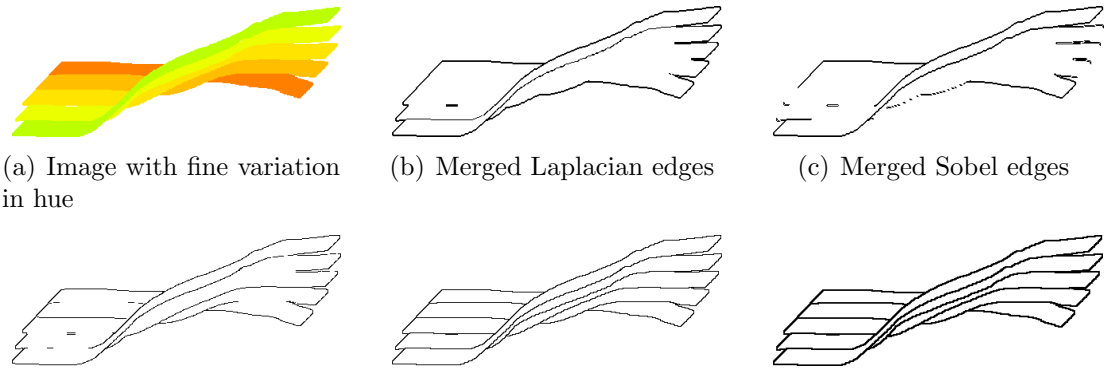
$$SA(\overrightarrow{v}, \overrightarrow{w}) = \cos^{-1}\left(\frac{\overrightarrow{v} \cdot \overrightarrow{w}}{\| \overrightarrow{v} \| \, \| \overrightarrow{w} \|} \right) \quad (2.22)$$

The resulting hyperspectral, omnidirectional edge detector is then represented as

$$\begin{aligned} e_{i,j} = \quad & \frac{1}{12} \left[SA(\overrightarrow{v}_{i-1,j-1}, \overrightarrow{v}_{i,j}) + 2SA(\overrightarrow{v}_{i,j-1}, \overrightarrow{v}_{i,j}) + SA(\overrightarrow{v}_{i+1,j-1}, \overrightarrow{v}_{i,j}) \right. \\ & + 2SA(\overrightarrow{v}_{i-1,j}, \overrightarrow{v}_{i,j}) + 2SA(\overrightarrow{v}_{i+1,j}, \overrightarrow{v}_{i,j}) + SA(\overrightarrow{v}_{i-1,j+1}, \overrightarrow{v}_{i,j}) \\ & \left. + 2SA(\overrightarrow{v}_{i,j+1}, \overrightarrow{v}_{i,j}) + SA(\overrightarrow{v}_{i+1,j+1}, \overrightarrow{v}_{i,j}) \right], \end{aligned} \quad (2.23)$$

where $e_{i,j}$ is the response of the edge detector for the central pixel $\overrightarrow{v}_{i,j}$. Figure 2.11(f) shows the result of this method.

From the results shown in Figure 2.11 (generated using the application developed during this research), it is clear that for an image with fine variations in the hue com-



(a) Image with fine variation in hue

(b) Merged Laplacian edges

(c) Merged Sobel edges

(d) Merged Canny edges

(e) Vector angle Roberts edges

(f) Vector angle modified Laplacian edges

Figure 2.11: Colour edge detection by various methods.

ponent, the vector angle based edge detectors outperform the plain edge detection methods. The different shades shown in Figure 2.11(a) have the hue component changing while the intensity and saturation remain the same. This emphasizes the importance of vector angle based edge detectors in the context of colour images.

2.3 Image Segmentation

One of the objectives of this research is to extend the existing skeletonization algorithm to colour images. In the case of a single object in an image, it is acceptable to rely on an edge detected image to define object boundaries. In the case of images with multiple objects, the situation is slightly more complex. Edges from such an image represent object boundaries, but there is no distinction between them. This means that there is no object definition attached with the edges and that they are global edges.

Therefore it is required to somehow define the objects first and then obtain the edges from them. This can be achieved using *image segmentation*. The main goal of image segmentation is to divide an image into parts that have a strong corre-

lation with objects or areas of the real world depicted in the image [Sonka et al., 1999, chap. 5]. Thus, image segmentation divides the whole image into homogeneous regions based on colour information. The regions can be loosely defined as representatives of objects present in the image. This section explores various segmentation algorithms and then explains the colour image segmentation algorithm adopted in this research.

Thresholding

Thresholding provides the simplest of all segmentation methods. A *threshold* value is used to segment objects and background from an image. The threshold value can be a *global threshold* or a *local threshold*. The most important task is to efficiently calculate the threshold value so that objects of interest are retained in the final image. Otsu [1979] provides a strategy to seek the optimal threshold value based on maximizing the gray-level variance between objects and background. For images with uneven illumination, a single threshold value is not sufficient. For such images, *adaptive thresholding* [Fisher et al., 1996] can provide good results. This is performed by subdividing the image into smaller regions and then utilizing different thresholds to segment each sub-image.

Region based segmentation

Region based segmentation methods construct the regions directly without actually finding a single colour value that separates them. In this category, *region growing* and *region splitting and merging* are the main methods.

Region growing [Sonka et al., 1999, chap. 5] is a procedure that groups smaller pixel regions into larger regions based on some criteria. The method starts with seed pixels in the image and keeps growing each seed based on a predefined criteria. The

procedure continues until no more pixels can be added.

Region splitting and merging [Horowitz and Pavlidis, 1974] is a top down approach for segmentation that begins with the entire image as one region. Based on brightness criteria, if a region is considered nonuniform then it is split into four quadrants. After splitting, each region thus created is compared with its adjacent region and merged if they are similar. The split and merge procedure is continued until the individual pixel level is reached.

Watershed segmentation

This segmentation method is based on the concept of watersheds. In topography, a watershed is an area of land that receives the rain and drains it to a water body that exists downhill [Gonzalez and Woods, 2002, chap. 10]. Each watershed is separated by a ridge or a mountain that forms the watershed line. A catchment basin is a point where the water drains. This method is based on visualizing the image in three dimensions as a topographic surface such that the gray levels represent altitudes. In a practical approach by Vincent and Soille [1991], the catchment basins are filled from the bottom. Each minimum gray level represents a catchment basin. Imagine that there is a hole at each local minima and that the whole surface is gradually immersed in water. As a result, water will start filling up the catchment basins whose minima are under the water level. If two catchment basins would start merging as a result of immersion, a dam is built all the way to the highest altitude. The dam acts as a separator between the two regions and finally comes out as a boundary between regions (see Figure 2.12).

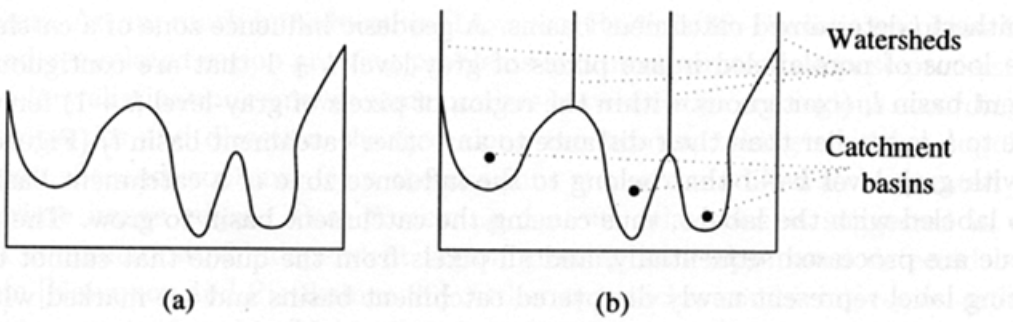


Figure 2.12: One dimensional example of watershed segmentation: (a) gray-level profile of image data; (b) watershed segmentation-local minima of gray-level (altitude) yield catchment basins, local maxima define the watershed lines. (from: Sonka et al. [1999])

Classification

The task of general image segmentation can be treated as an example of a classification problem [Russ, 2002, chap. 6]. Image classification, used extensively with satellite imagery, is a general approach to subdivide an image into meaningful regions. The main objective in a classification problem is to locate clusters belonging to various classes. These clusters should be well separated from each other. Existing methods of classification can be broadly grouped into *supervised* and *unsupervised* methods.

Supervised classification techniques require a priori knowledge of the cluster signatures. This is generally provided as training sets by human assistance [Russ, 2002, chap. 10]. A few methods in this category are: *Minimum-Distance-to-Means*, *Parallelopiped* Classifier, and *Maximum Likelihood*.

On the other hand, **unsupervised classification** methods attempt to define class regions in order to maximize the similarity of objects within each cluster and differences between classes [Russ, 2002, chap. 10]. *K-Means*, *Fuzzy C Means*, and *Isodata* classifiers are a few to mention in this category. Generally these methods

start by taking random seeds from the image. These seeds provide class signatures. Based on these signatures, the number of classes is defined and classification is carried out.

The segmentation method adopted here is the one provided by Comaniciu and Meer [1997] which is based on feature space analysis. This is explained next in detail.

2.4 Image Segmentation by Mean Shift Algorithm

Feature space analysis is used extensively in image understanding tasks. Comaniciu and Meer [1997] provide a comparatively new and efficient segmentation algorithm that is based on feature space analysis and relies on the *mean-shift algorithm* to robustly determine the cluster means. A *feature space* is a space of feature vectors. These features can be object descriptors or patterns in case of an image. As an example, if we consider a colour image having three bands (red, green, and blue) then the image we see as intensity values plotted in Euclidean XY space is said to be in *image space*. Consider a three dimensional space with the axes being the three bands of the image. Each colour vector corresponding to a pixel from the image can be represented as point in the feature space.

2.4.1 Mean Shift Algorithm

Given n data points x_i, $i = 1, \ldots, n$ in the d-dimensional space R^d, a *flat kernel* that is characteristic function of the λ-ball in R^d is defined as

$$K(x) = \begin{cases} 1 \text{ if } \|x\| \leq \lambda \\ 0 \text{ if } \|x\| > \lambda \end{cases} \tag{2.24}$$

The *mean shift* vector at a location x is defined as

$$M_\lambda(x) = \frac{\sum_{r \in R^d} xK(r-x)}{\sum_{r \in R^d} K(r-x)} - x \tag{2.25}$$

Cheng [1995] shows that the mean shift vector, the vector of difference between the local mean and the center of the window $K(x)$, is proportional to the gradient of the probability density at x. Thus mean shift is the steepest ascent with a varying step size that is the magnitude of the gradient. Comaniciu and Meer [2002] use mean shift vector in seeking the mode of a density by shifting the kernel window by the magnitude of the mean shift vector repeatedly. The authors also prove that the mean shift vector converges to zero and eventually reaches the basin of attraction of that mode. This is shown graphically in Figure 2.13 where the initial position of the kernel window is chosen at a location far from the mode of the data and the window is moved by a magnitude equal to that of the mean shift vector at the current window location in each step. It can be seen that the mean shift vectors gradually converge as the window moves near the maximum density region.

Comaniciu and Meer [1997] state a simple, adaptive steepest ascent mode seeking algorithm.

1. Choose the radius r of the search window (i.e, radius of the kernel).
2. Choose the initial location of the window.
3. Compute the mean shift vector and translate the search window by that amount.
4. Repeat till convergence.

The mean shift algorithm gives a general technique of clustering multi-dimensional data and is applied here in colour image segmentation. The fundamental use of mean

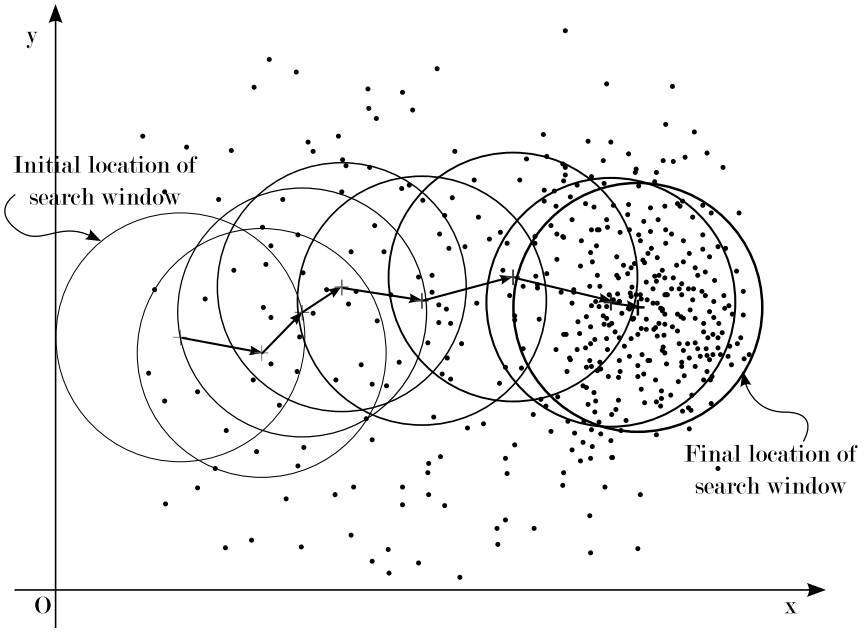


Figure 2.13: Mode seeking using mean shift algorithm.

shift is in seeking the modes that give regions of high density in any data.

2.4.2 Application to Image Segmentation

Comaniciu and Meer [1997] provide an autonomous segmentation technique with only the type of segmentation to be specified by the user. This method emphasizes the importance of utilizing the image space along with the feature space to efficiently perform the task of segmentation. The segmentation has three characteristic input parameters:

- Radius of the search window, r,
- Smallest number of elements required for a significant colour, N_{min}, and
- Smallest number of connected pixels necessary for a significant image region, N_{con}.

The size of the search window determines the resolution of the segmentation, smaller values corresponding to higher resolutions. The authors use square root of the trace of global covariance matrix of the image, σ, as a measure of the visual activity in the image. The radius r is taken proportional to σ. For the implementation of the segmentation algorithm, the authors provide three segmentation resolution classes:

1. **Undersegmentation** refers to the lowest resolution with a minimum number of colours and only dominant regions of the image. The three parameters for this class are: $(0.4\sigma, 400, 10)$.

2. **Oversegmentation** refers to intermediate resolution and represents objects with some level of detail. The three parameters for this class are: $(0.3\sigma, 100, 10)$.

3. **Quantization** refers to the highest resolution and produces image with all the important colours with no object connectivity requirement. The three parameters for this class are: $(0.2\sigma, 50, 0)$.

The following is an excerpt from [Comaniciu and Meer, 1997] stating the segmentation algorithm using feature analysis and mean shift algorithm.

1. **Definition of the segmentation parameters.**

   The parameters r, N_{min} and N_{con} are chosen in this step. The size of the search window determines the resolution of the segmentation with smaller values corresponding to higher resolutions and vice versa. The authors choose the value of r based on the square root of the trace, σ, of the global covariance matrix of the image which is proportional to the power of the signal (image). The other two parameters are defined as per the chosen class of segmentation.

2. **Definition of the search window.**

   The initial location of the search window is randomly chosen. In order to ensure

that the search starts near a high density region, several location candidates are examined and the region containing the highest density of feature vectors is selected.

3. **Mean shift algorithm.**

   The closest mode is located using the mean shift algorithm applied to the selected search window. Convergence is declared when the magnitude of the shift becomes less than 0.1 units in the feature space.

4. **Removal of detected feature.** The pixels yielding feature vectors inside the search window at its final location are discarded from both the image domain as well as the feature space. Their 8-connected neighbors are also removed in order to clean the background.

5. **Iterations.**

   Steps 2 to 4 are repeated until no further feature vectors can be found with N_{min} elements.

6. **Determining initial feature palette.**

   In the feature space a significant colour must be based on minimum N_{min} vectors, while in the image domain more than N_{min} pixels of that colour should belong to a connected component. From the extracted colours only those are retained for the initial feature palette that yield at least one connected component in the image of size larger than N_{min}. The 8-neighbors removed in step 4 are also considered in defining the connected components.

7. **Determining the final feature palette.**

   All the pixels are reallocated based on the initial feature palette. First, the pixels yielding feature vectors inside the search windows at their final location

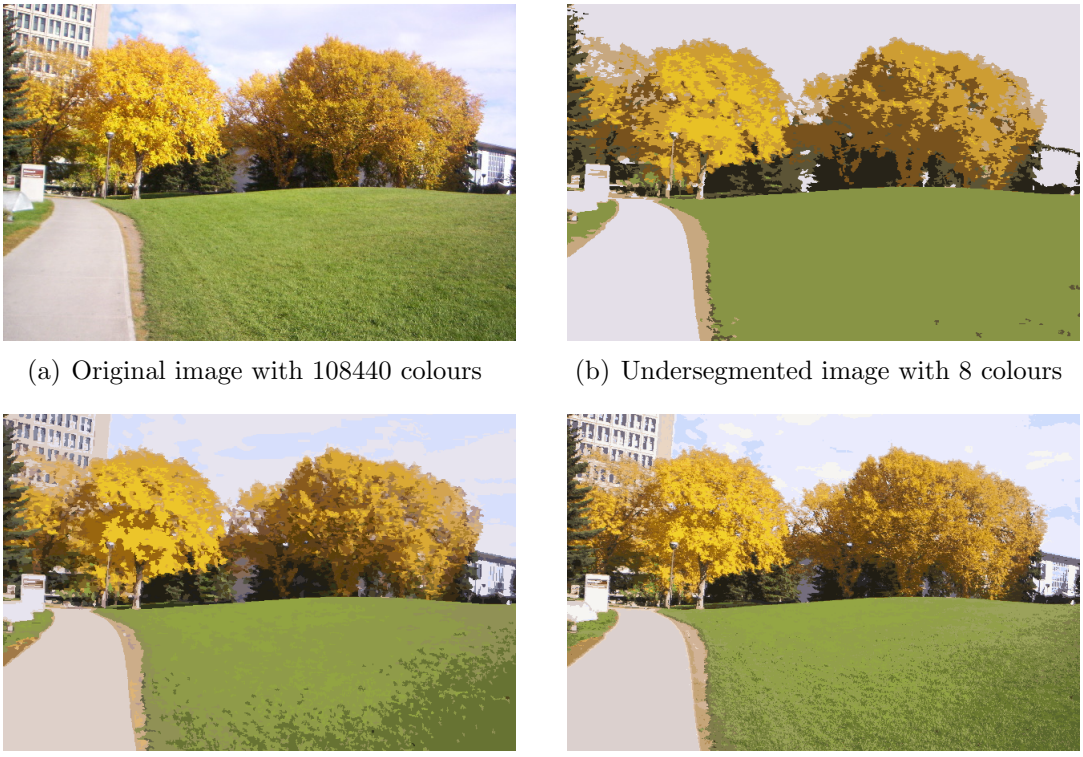
are considered. These pixels are allocated to the colour of their respective window centres without considering image domain information. The windows are then inflated to double volume by multiplying the radius by $\sqrt[3]{2}$. The newly incorporated pixels are retained only if they have at least one neighbour which was already allocated to that colour. The mean of the feature vectors mapped into the same colour is the value retained for the final palette. A few unclassified pixels left at the end of this process are allocated to the closest colour in the final feature palette.

8. **Postprocessing.**

   This step removes all small connected components of size less than N_{con}. These pixels are allocated to the majority colour in their 3×3 neighbourhood, or in case of a tie to the closest colour in the feature space.

Figure 2.14 shows the results of this segmentation algorithm on a natural image. Note the variation in number of colours for each segmentation type.

Later, Comaniciu and Meer [2002] provide an improvement over this segmentation algorithm by merging the image domain and the feature (range) space into a joint spatial-range domain of dimension $d = p + 2$, where p is the dimension of the range domain. This gives an added advantage of considering both the spaces together and gives good results in cases where non-uniform illumination produces false contours when the previous segmentation algorithm is used. Therefore, the new algorithm is particularly useful to segment natural images with man-made objects. An added computational overhead to process higher dimensional space is inevitable here. In this research, since we are dealing with scanned maps, the simple mean shift based segmentation algorithm provides satisfactory results.



(a) Original image with 108440 colours

(b) Undersegmented image with 8 colours

(c) Oversegmented image with 34 colours

(d) Quantized image with 49 colours

Figure 2.14: Colour image segmentation by Comaniciu and Meer [1997]

Segmentation provides us with definite boundaries of objects that are used to extract sampling points around an object. Discussed next are the Delaunay triangulation and Voronoi diagram based skeletonization methods. It is first important to understand the underlying data structure used in this research for the implementation of the skeletonization algorithm. This data structure known as the *quad-edge* data structure is the next topic of discussion.

2.5 The Quad-edge Data Structure

In order to store and work on the Delaunay triangulation and the Voronoi diagram of the set of sampled points around an object, we need an efficient data structure

that enables easy navigation through these graphs and facilitates computation of the skeleton. The quad-edge data structure [Guibas and Stolfi, 1985] is an elegant and natural storage structure for graphs. It can simultaneously represent a graph and its dual. The edges of the graph are directed and there are four such edges representing two symmetric edges from both the graph and its dual.

For any oriented and directed edge e, Guibas and Stolfi [1985] define its vertex of *origin*, $e.Org$, its *destination*, $e.Dest$, its *left face*, $e.Left$, and its *right face*, $e.Right$. The *symmetric* edge of e, $e.Sym$ is defined as the same edge but with *opposite direction*. An edge $e.Rot$ is called the counter-clockwise *rotated* version of e.

As shown in Figure 2.15, a quad-edge record is composed of four directed edges starting from edge e. It also shows a sample graph and its representation in the quad-edge data structure. The group of edges containing e is represented in the data structure by one *edgerecord* e, divided into four parts e[0] through e[3]. Each successive edge in a quad-edge record is a rotated version of the previous edge. Thus, part e[r] corresponds to the edge $e_0 Rot^r$. Every edge has its origin coordinate and a reference to the next counterclockwise edge sharing the same origin. Therefore, there are two groups of edges in a quad-edge record. One contains two symmetric edges of the primal graph (e and $e.Sym()$) and the other contains two symmetric edges of the dual graph ($e.Rot()$ and $e.Rot().Sym()$). Navigation from a primal edge to its dual edge is achieved by the $Rot()$ operator that gives the next counter-clockwise rotated edge.

As discussed by Guibas and Stolfi [1985], an edge e in a group of edges from a particular quad-edge record can be represented as $e_0 Rot^r$, where $r \in \{0, 1, 2, 3\}$ and e_0 is the canonical representative of the group to which e belongs. A generic edge $e = e_0 Rot^r$ can be represented by the duplet (e, r). To navigate through the graph and its dual in a quad-edge data structure, we require two basic (atomic) edge

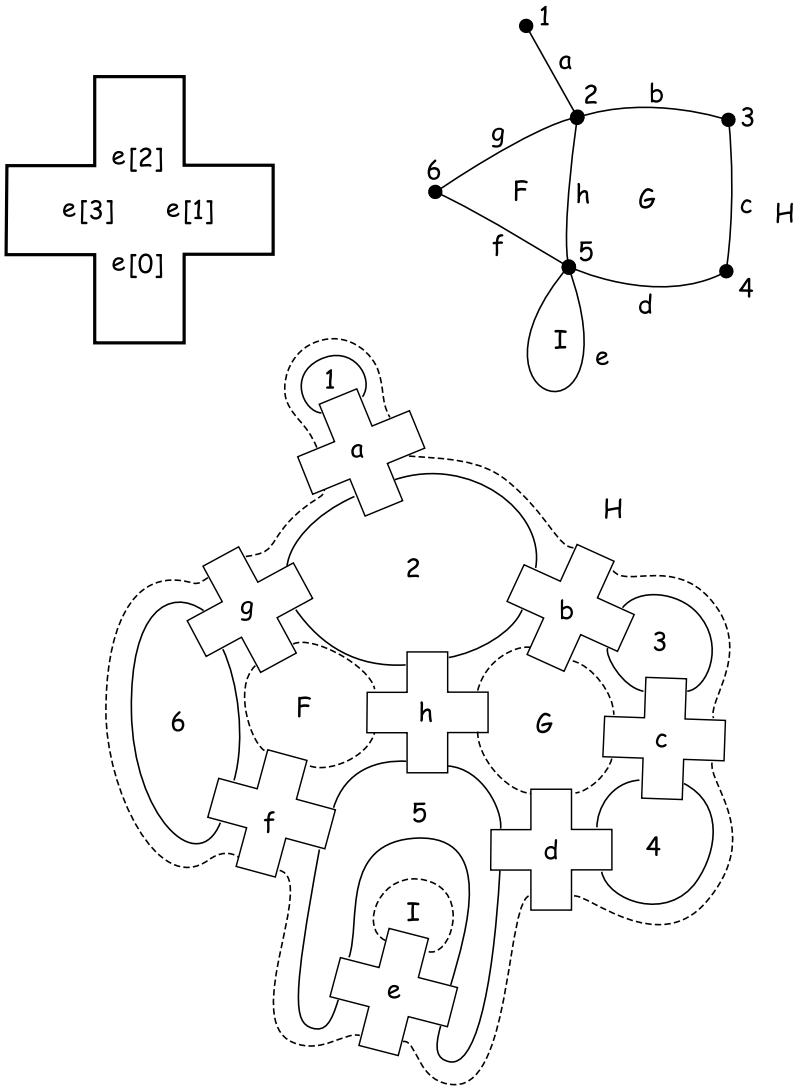


Figure 2.15: Quad-edge record showing Next links.

functions *Rot* and *Onext* given by the formulas

$$Rot\,(e,r) = (e, r+1) \tag{2.26}$$

$$Onext\,(e,r) = (e\,[r]\,.Next) \tag{2.27}$$

From these formulas, we can define a few more functions

$$Sym\,(e,r) = (e, r+2) \tag{2.28}$$

$$Rot^{-1}\,(e,r) = (e, r+3) \tag{2.29}$$

$$Oprev\,(e,r) = Rot\,(e\,[r+1]\,.Next) \tag{2.30}$$

In these formulas, $Rot\,(e,r)$ refers to the dual edge that is the counter-clockwise rotated edge to (e,r). $(e,r)\,Onext$ refers to the next counter-clockwise edge that shares the same origin with (e,r). $Sym\,(e,r)$ is the edge symmetric to (e,r). $Rot^{-1}\,(e,r)$ refers to the dual edge that is the clockwise rotated edge to (e,r). Lastly, $Oprev\,(e,r)$ is the next clockwise edge that shares the same origin with (e,r). Figure 2.16 graphically shows these edge functions.

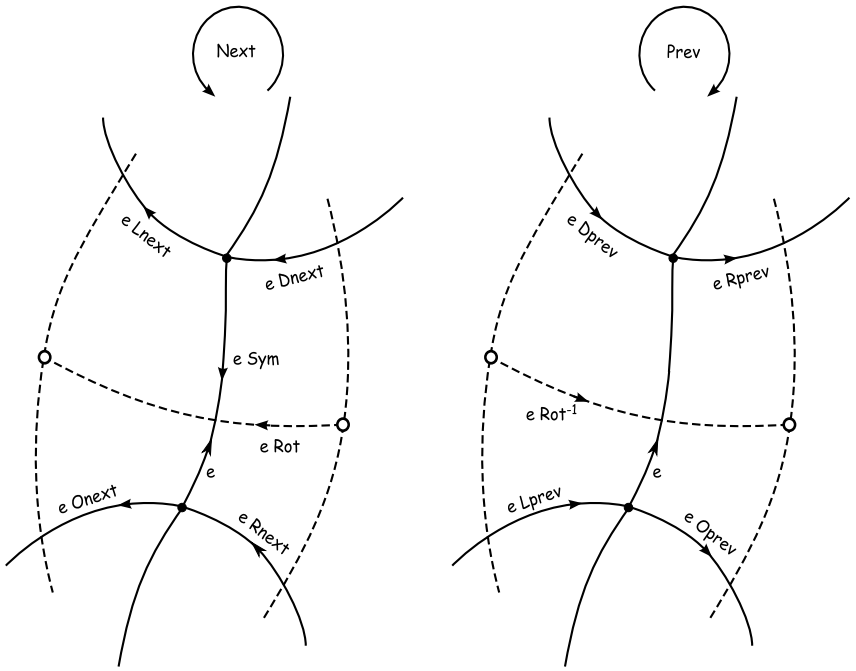


Figure 2.16: Various edge functions in the quad-edge data structure.

For our purpose, where we need to store the Delaunay triangulation and the

Voronoi diagram, the quad-edge data structure proves to be very convenient. While the Delaunay graph is stored in the primal edges in our data structure, the Voronoi graph is stored in the dual edges. Interestingly, the computation of the Delaunay triangulation leads to the computation of the Voronoi diagram. Only coordinate values of the Voronoi vertices are calculated afterwards; the topology is maintained by the data structure. Next, we will see an incremental algorithm for the computation of the Delaunay triangulation.

2.5.1 Incremental Construction of Delaunay Triangulation

An excellent comparison of algorithms to compute the Delaunay triangulation is given by Su and Drysdale [1995]. The authors classify sequential algorithms to compute the Delaunay triangulation in five categories:

1. *Divide-and-conquer* algorithm that work by dividing the original point-set into smaller subsets and solving the problem recursively. The smaller triangulations are then merged together. Guibas and Stolfi [1985] discuss a divide and conquer algorithm in detail and also present its implementation in the quad-edge data structure. Dwyer [1987] also provides a faster divide-and-conquer algorithm for the same.

2. *Sweepline* algorithm in by Fortune [1987] is an optimal scheme to construct Delaunay triangulation. A sweep line in one direction constructs the triangulation as it encounters the sites.

3. *Incremental* construction algorithms add sites one by one and update the diagram after each site is added. Green and Sibson [1977] present a basic incremental algorithm. Guibas et al. [1990] propose a better incremental algorithm with simpler analysis of its runtime.

4. Yet another method to compute Delaunay triangulation is to grow a triangle at a time in a manner similar to *gift wrapping* algorithms for convex hulls [Dwyer, 1989; Maus, 1984; Tanemura et al., 1983]. Here, the Delaunay triangulation is constructed by starting with a single Delaunay triangle and then incrementally discovering valid Delaunay triangles, one at a time. Each new triangle is grown from an edge of a previously discovered triangle by finding the site that joins with the endpoints of that edge to form a new triangle whose circumcircle is empty of sites [Su and Drysdale, 1995].

5. Lifting the sites into three dimensions and computing their *convex hull* [Barber, 1993] computes the Delaunay triangulation using a very basic property discussed in subsection 1.5.6.

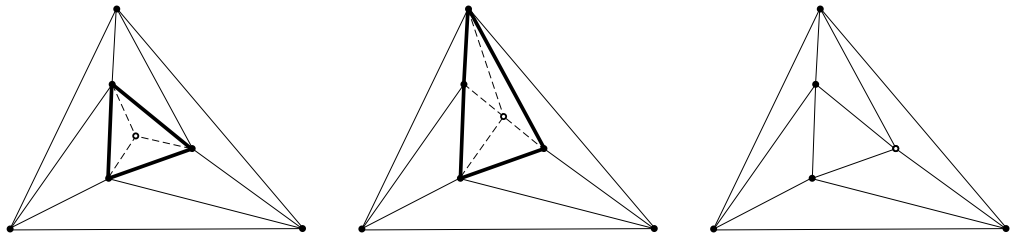
The incremental algorithm provided by Green and Sibson [1977] is discussed here. An elegant implementation has been provided in [Guibas and Stolfi, 1985]. For definition of the Delaunay triangulation, refer to subsection 1.5.6 on page 13. We start with the assumption that all sites in the set strictly fall inside a large triangle whose vertices are considered to be among the given sites. The Delaunay diagram is computed incrementally by inserting one site at a time to the existing triangulation. This is done by first locating the triangle in which the new site will be contained. There are three cases that arise here:

1. The new site falls inside a triangle in the triangulation (see Figure 2.17(a)). In this case, three new edges are added to the triangulation. These edges connect the vertices of the containing triangle to the new site.

2. The new site falls on an edge in the triangulation (see Figure 2.17(b)). In this case, the edge on which the site falls is deleted and the four new edges are added

to the triangulation. These edges connect the vertices of the quadrilateral created after deleting an edge to the new site.

3. The new site falls on an existing site in the triangulation (see Figure 2.17(c)). In this case we ignore the new site.



(a) Case I: Inside a triangle (b) Case II: On an edge (c) Case III: On a site

Figure 2.17: Three possibilities for a new site to fall within an existing triangulation.

All the new edges incident to the new site are guaranteed to be Delaunay edges. The edges of the triangle or the quadrilateral containing the new site might not be Delaunay. Therefore, all such suspect edges (shown as thick lines in Figure 2.17) are tested for the Delaunay condition using the incircle test (see subsection 1.5.6 on page 13). The test is carried out on the four sites belonging to the two triangles on either side of the suspect edge. The new site is always one among these. If the edge fails the incircle test then it is replaced by the other diagonal of its quadrilateral. The new diagonal edge is a confirmed Delaunay edge. Diagonal edge swapping is shown in Figure 2.18 where the diagonal shown as a dashed line is replaced by the other diagonal to satisfy the Delaunay criterion. The whole procedure is shown in Algorithm 1.

Algorithm 1: Incremental algorithm to compute the Delaunay triangulation

input : Set of sites S
output: Delaunay triangulation $\mathscr{D}$ of S

```
1  begin
2      Consider initial triangulation D composed of a triangle such that all sites
       in S fall inside it;
3      foreach site s in S do
4          e ←Locate(s);
           /* Returns edge e so that s is immidiately to its right   */
5          if s = e.Origin Or s = e.Destination then return; /* Ignore it */
6          else if s is on e then DeleteEdge(e);
           /* Connect s to vertices around it                        */
7          foreach edge e surrounding s do
8              Store e in E_suspect;
9              Create an edge joining s to e.Origin and e.Destination;
10         end
11         foreach edge e in E_suspect do
12             t ← site opposite to s belonging to the triangle formed with e;
13             if InCircle(e.Origin, t, e.Destination, s) then
14                 Replace e with an edge joining s and t;
15             end
16         end
17     end
18 end
```

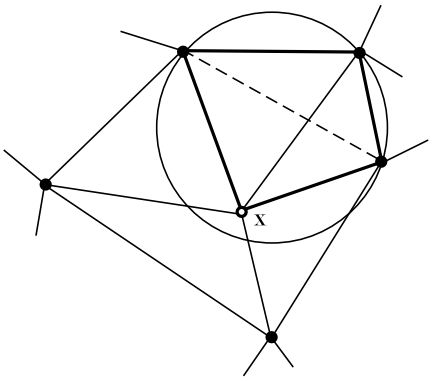


Figure 2.18: Swapping a suspect edge.

2.5.2 Computation of the Voronoi Diagram

Once the Delaunay triangulation is constructed, it is very easy to get the Voronoi Diagram. Since the quad-edge data structure allows simultaneous storage of both of these graphs, we have the Voronoi diagram already computed along with the Delaunay graph, only coordinates of the Voronoi vertices are required to be calculated. This involves computation of circumcenters of all the triangles in the Delaunay triangulation. For arbitrary origin O, the circumcenter X of a triangle ABC in Figure 2.19 is given by:

$$\overrightarrow{OX} = \overrightarrow{OM} + \frac{1}{2}\left(\frac{\vec{v_a} \cdot \vec{v_b}}{\vec{n} \cdot \vec{v_b}}\right)\vec{n} \tag{2.31}$$

where,

$$\vec{n} = \begin{bmatrix} -\overrightarrow{v_{c_y}} \\ \overrightarrow{v_{c_x}} \end{bmatrix}, \text{ and}$$

$$\overrightarrow{OM} = \frac{\overrightarrow{OA} + \overrightarrow{OB}}{2}$$

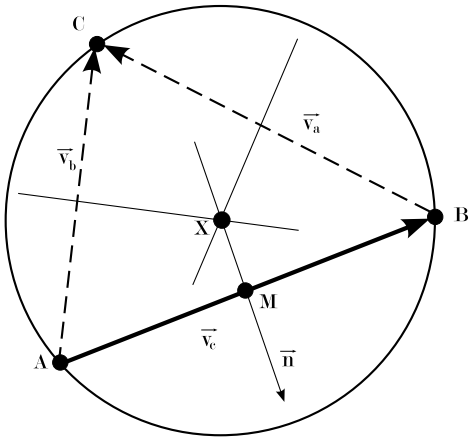


Figure 2.19: Voronoi vertex.

2.6 Crust Extraction by Gold

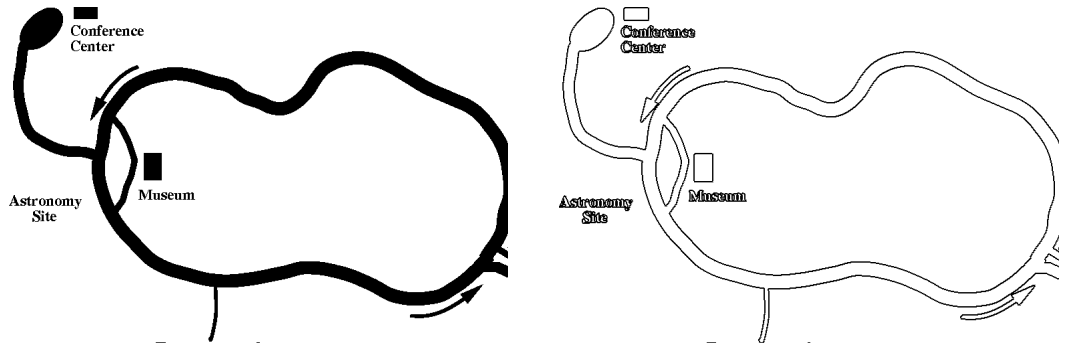
Work by Amenta et al. [1998] leads to the extraction of an object boundary from a set of sufficiently well sampled data points. The vertices of the Voronoi diagram approximate the medial axis of a set of sample points from a smooth curve. In their research, vertices of the Voronoi diagram of the sample points were inserted into the original set of sample points and a new Delaunay triangulation was computed. The circumcircles of this new triangulation approximate empty circles between the original boundary of the object and its skeleton. Thus, they conclude that any Delaunay edge connecting a pair of the original sample points in the new triangulation is a part of the border.

Further research by Gold [1999] leads to a One-step border (crust) extraction algorithm. In a Delaunay triangulation, each Delaunay edge is adjacent to two triangles and the circumcircles of these triangles are the Voronoi vertices. A Voronoi edge connecting these two circumcenters is the dual edge to the Delaunay edge considered here. According to Gold [1999], a Delaunay edge is a part of the border if it has a circle that does not contain any Voronoi vertex. It is sufficient to test

only the vertices of the dual Voronoi edge. The test is the standard $InCircle$ test. Considering two triangles (p, q, r) and (r, q, s) sharing an edge (q, r) in a Delaunay triangulation and let v be the a vector orthogonal to edge $(r - q)$ in clockwise order, then the test becomes:

$$(s - q) \cdot (s - r) * (p - q) \cdot (p - r) \geq -(s - r) \cdot v * (p - q) \cdot v \tag{2.32}$$

This test will be true for an edge in the border set. Furthermore, those Delaunay edges that are not the part of the border set have their dual Voronoi edges as being part of the skeleton (shown in Figure 2.20).



(a) Map (b) Boundary

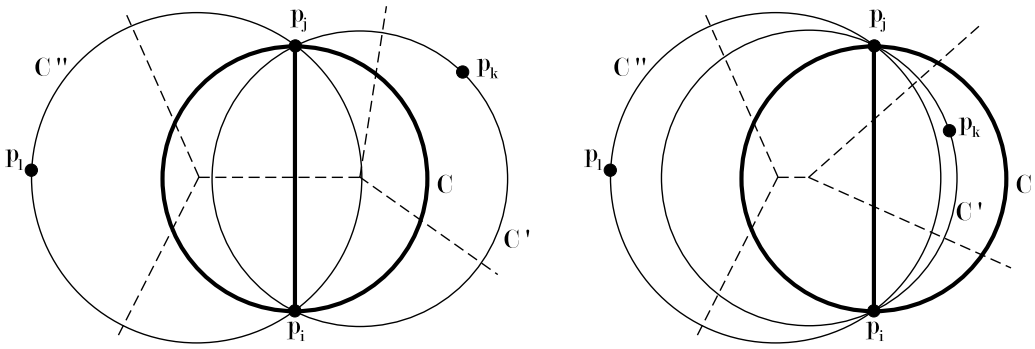
Figure 2.20: Result of boundary (crust) extraction using algorithm by Gold.

2.7 Skeleton Extraction by Anton et al.

Research by Anton et al. [2001] suggests a new algorithm for skeleton extraction. This is based the concept of *Gabriel Graph* [Gabriel and Sokal, 1969].

A Gabriel graph G (highlighted in Figure 2.22) is a connected subset of the Delaunay graph $\mathscr{D}$ of points in set S such that two points p_i and p_j in S are connected by an edge of the Gabriel graph if and only if the circle with diameter $p_i p_j$ does not

contain any other point of S in its interior. In other words, the edges in G are those edges from $\mathscr{D}$ whose dual Voronoi edges intersect with them. Consider four points $p_i, p_j, p_k,$ and p_l on a plane. Figure 2.21(a) shows Gabriel edge p_ip_j that satisfies the empty circle criterion. Circles C' and C'' are circumcircles of triangles $p_jp_ip_k$ and $p_ip_jp_l$ respectively and circle C is a circle with diameter as edge p_ip_j. Figure 2.21(b) shows the case when the Gabriel condition is not met.



(a) A Gabriel edge (b) Not a Gabriel edge

Figure 2.21: Condition for a Gabriel edge.

Given the Delaunay triangulation $\mathscr{D}$ and the Voronoi diagram V of sample points S from the boundary of an object, the algorithm for centreline extraction in [Anton et al., 2001] proceeds by selecting all the Gabriel edges in graph G. Each dual Voronoi edge v of the Gabriel edge g from G is inserted in the skeleton K if the following

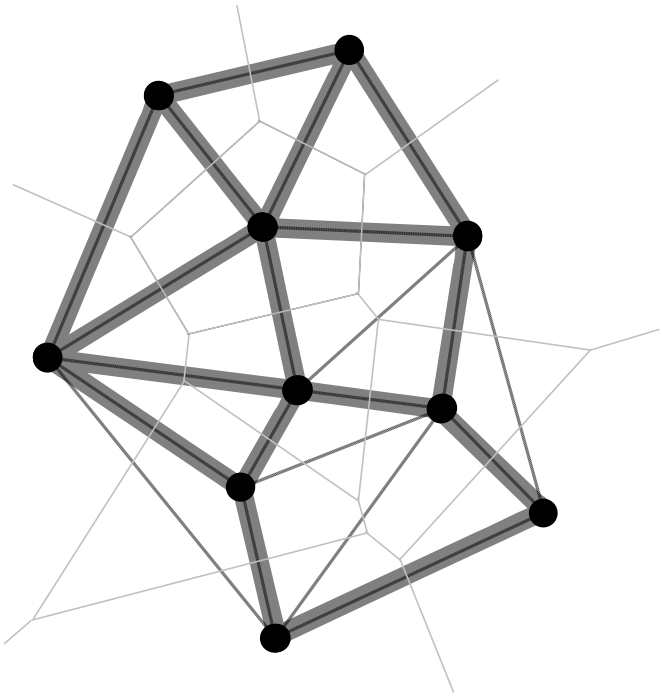


Figure 2.22: Gabriel graph highlighted in a Delaunay triangulation.

condition is met:

$$g.Origin.Colour \neq g.Destination.Colour$$

Or

$$g.Origin.Colour \neq v.Origin.Colour$$

Or

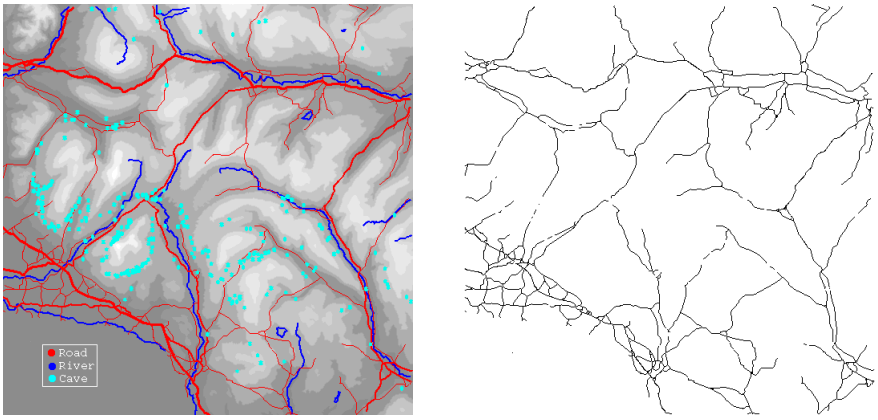
$$g.Origin.Colour \neq v.Destination.Colour$$

And

$$\|g.Origin.Colour - g.Dest.Colour\| \geq \|v.Origin.Colour - v.Dest.Colour\| \tag{2.33}$$

Here, *Origin.Colour* and *Destination.Colour* are colour values from the gray scale image corresponding to the location of the origin and the destination of an edge respectively. Figure 2.23(b) shows the result of skeleton extraction from streams

present in a map (Figure 2.23(a)). However, a close observation reveals that the skeleton thus obtained has gaps. These gaps are prominent if the object under consideration has sharp turns in its geometry which is further amplified if the object is thick. Ongoing research by Anton et al. [2001] tries to overcome this by locating skeleton edges by tracing the skeleton along the border set.



(a) Map (b) Skeletons of streams in red

Figure 2.23: Result of skeleton extraction using the algorithm by Anton et al.

This chapter covered the previous research done in the field of feature extraction. Basic definition of skeletonization and popular methods for its computation were reviewed. Concepts of edge detection and colour image segmentation were explored to define object boundaries in a digital image. Delaunay triangulation and Voronoi diagram based methods to extract the boundary and the skeleton of objects were discussed. This research work continues the work by Anton et al. [2001] further to process colour scanned maps. A strategy for the same is presented in the next chapter.

Chapter 3

Extension of Existing Algorithm to Colour Images

Colour images provide more contextual details about the objects present in the image. Therefore, processing colour images rather than gray scale images can provide much more accurate information. The skeleton extraction algorithm by Anton et al. [2001] processes only gray scale images. One of the main objectives of this research is to extend this algorithm to process colour images as well.

3.1 Considerations to Process Colour Information

The tremendous amount of information associated with colour images should be judiciously used in order to get expected results. Following are a few points in this direction:

- **Object definition**. Objects are defined better in colour images and therefore ambiguity in object boundary definition can be easily resolved [Gonzalez and Woods, 2002, chap. 6]. Hue of the colour should be used along with the intensity

of the colour to discern objects accurately.

- **Global edges versus object edges**. An edge detection algorithm will detect main edges but without any object specific information. These edges are actually global edges which do not carry any information about object boundaries. If object specific information has to be extracted, like skeletons, then object edges should be detected from the image.

- **Bounded objects**. For skeleton extraction, a bounded object will make things simpler. Trying to define objects using simple edge detection might result in open objects since the edges may not form a closed curve.

In light of the above points, it can be concluded that there is a need for object based distinction in edges, especially when more objects are present in the image. Initial experiments done with colour edge detection helped in deducing the fact that object delineation via segmentation handles the information in a better way in light of the points discussed above. The general approach adopted is:

1. Segment a colour image into prominent objects.

2. Ask the user to select an object or process all the objects independently.

3. Collect sample points for each object to be processed.

4. Extract the skeletons using Delaunay/Voronoi diagram based algorithm.

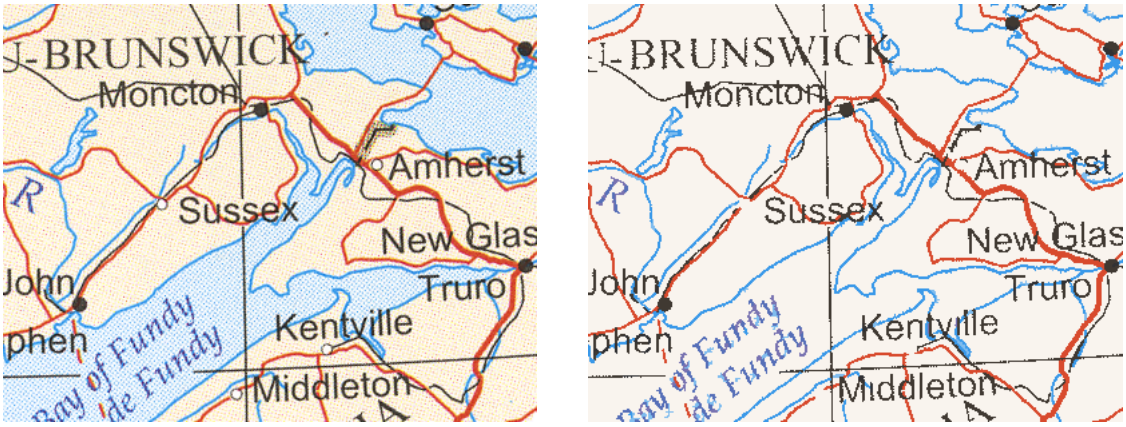
With this framework in mind, the next sections will explore various steps in detail. The method discussed above and the algorithms discussed here have been implemented in an application named **VGUI** (Voronoi GUI). This is an application with a GUI (Graphical User Interface) that makes use of the platform independent **Qt** GUI framework (library). All of the code has been written in standard C++ on a 2.8

GHz Intel machine running SUSE Linux 10.0 (OSS) operating system. Figure 3.8 shows the application front end.

3.2 Colour Image Segmentation

Segmentation aims at partitioning an image into contiguous regions. Section 2.4 presented an overview of an efficient colour image segmentation algorithm proposed by Comaniciu and Meer [1997]. Following are the two main advantages of using it:

1. It provides robust recovery of significant image features that can be used in a content-based query system. Figure 3.1(b) shows the result of oversegmentation of the map in Figure 3.1(a). Note that complex background textures are removed after segmentation.



(a) Map (b) Map after segmentation

Figure 3.1: Detection of significant features using segmentation.

2. *Anti-aliasing* [Foley et al., 1990] is a technique in computer graphics of introducing fuzzy edges around an object in order to remove sampling artifacts also known as *aliasing*. Aliasing may result in *the jaggies* or *staircasing* in a raster

image. Since anti-aliasing introduces intermediate shades of a colour around objects making it difficult to define a proper object boundary, it is undesirable in our case. Undersegmentation removes the anti-aliasing effect. Figure 3.2(b) shows an underegmented image from Figure 3.2(a).

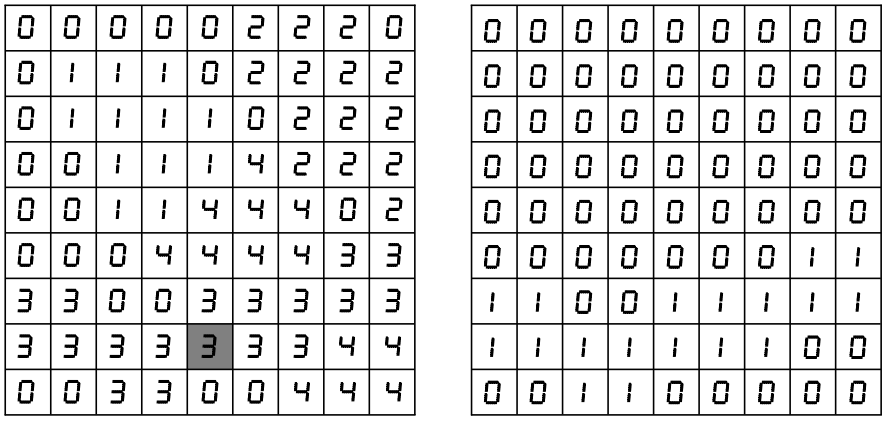
(a) Portion of maple leaf (b) Undersegmented image

Figure 3.2: Removal of anti-aliasing present in scanned images.

3.3 Object Selection from Segmented Image

Once objects are defined as homogeneous regions by the segmenter, the next step is to select them and operate on them. This is implemented as interactive object selection in the application VGUI. To achieve this, the user is allowed to click in a region on the image. If an object is composed of more than one region then multiple object selection can be made and regions combined to form a single object (*Shift + Click*). A wrongly selected region can be removed from the selection (*Ctrl + Click*). The user input is processed and the selected region is highlighted and selected for next processing.

Figure 3.3 shows the selection mechanism. Consider that the matrix shown in Figure 3.3(a) represents the segmented image with four objects labeled as $1, 2, 3$, and 4. The background is labeled as 0. If the user selects a pixel at the 8^{th} row and 5^{th}



(a) Segmentated image (b) Selected object

Figure 3.3: Object selection.

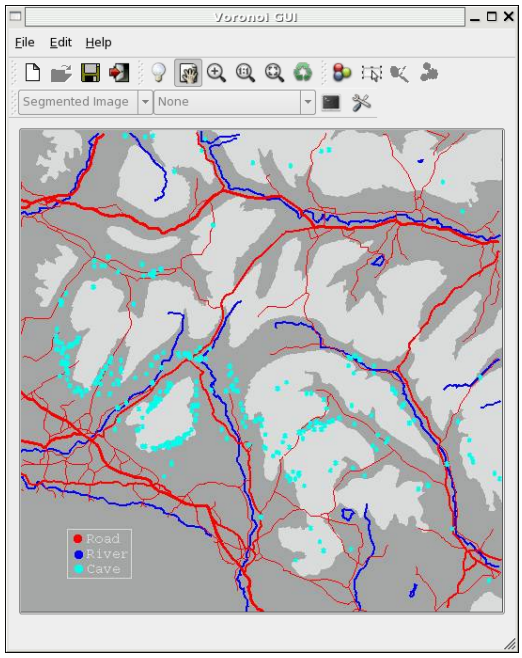
column, then the object labeled 3 is selected. A binary image corresponding to the selected object is built as shown in the matrix in Figure 3.3(b). Further processing is done to collect sample points along the boundary of the selected object.

As an example, consider the segmented image shown in Figure 3.4(a). The selected object is highlighted in black color (see Figure 3.4(b) and (c)). The binary image of the selected object can also be displayed.

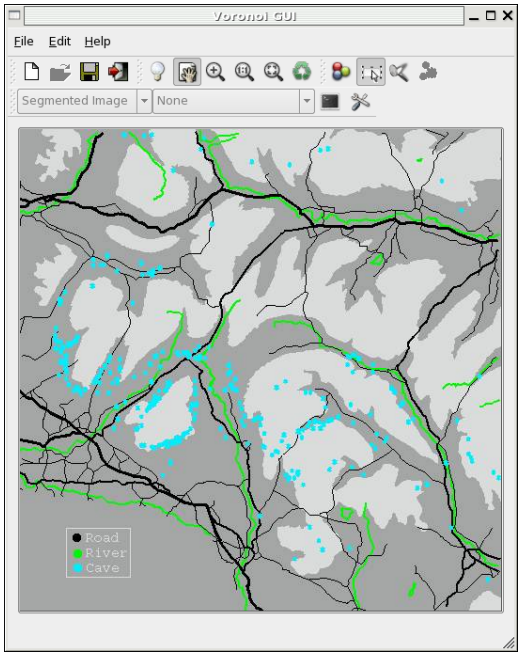
3.4 Object Sample Points Collection

Once we have an object chosen from an image, the next step is to sample its boundary in order to generate points used to construct the Delaunay triangulation. In order to automatically generate these sample points, edge pixels that are returned by the morphological edge detector are used. Using edge pixels also helps in generating a dense sampling which is required to give a better approximation of the skeleton [Amenta et al., 1998].

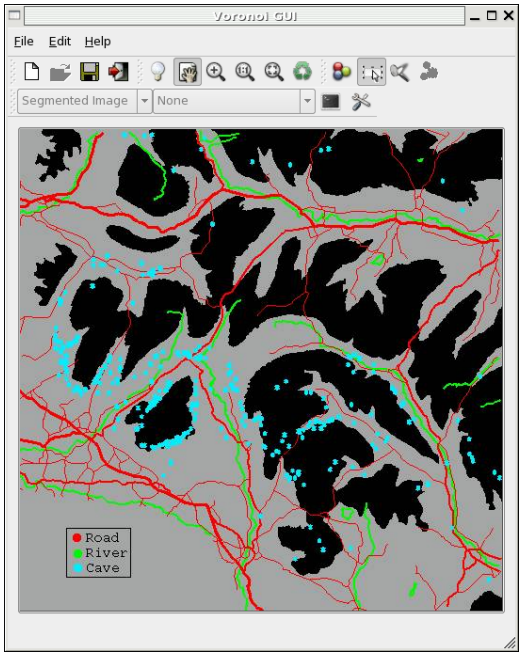
Morphological edge detection on the binary image containing the selected object



(a) Segmented image



(b) Selected linear object

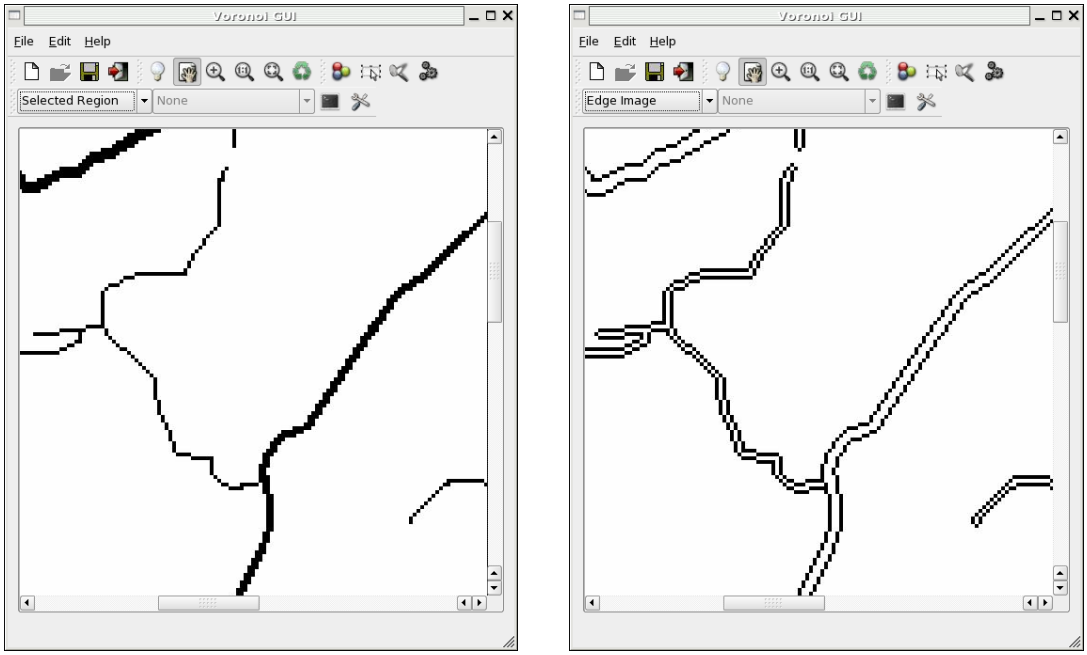


(c) Selected areal object

Figure 3.4: Example of object selection.

is performed as described in subsection 2.2.1. If the object is a linear feature then equation 2.14 is used to compute its edge pixels, otherwise equation 2.13 is used.

The edge pixels are then sequentially inserted into the Delaunay triangulation. The triangulation is updated after every insertion (using the incremental algorithm). Figure 3.5 shows object sampling.



(a) Object binary image (b) Sample points (edges)

Figure 3.5: Sampling along boundary of an object.

3.5 Boundary and skeleton extraction

Colour image processing, object selection and sampling are the three processes for the preparation of boundary and skeleton extraction. These are integrated into the existing application. The Delaunay triangulation of the sample points (see Figure 3.6(a)) is computed using the incremental algorithm given by Green and Sibson [1977] which is stored in the quad-edge data structure. This is followed by compu-

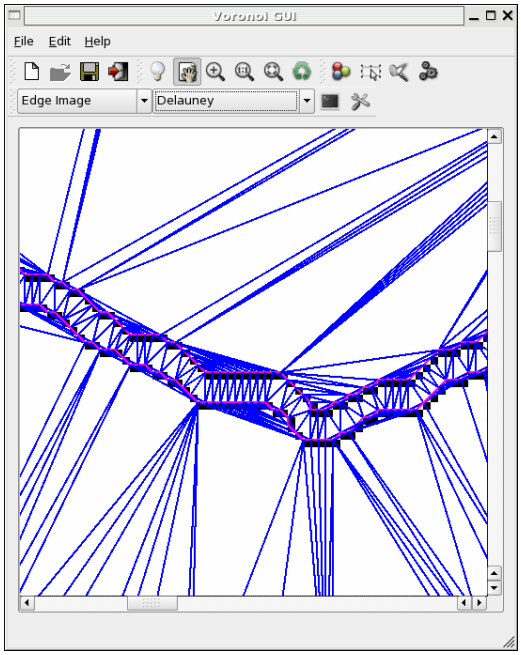
tation of the Voronoi vertices for all faces of the triangulation (see Figure 3.6(b)). The boundary of the object is extracted using the criteria given by Gold [1999] (see section 2.6) and the skeleton is extracted using the method based on Gabriel edges as suggested by Anton et al. [2001] (see section 2.7).

The edges in the Delaunay and Voronoi graphs are analyzed and flagged as being part of the boundary and the skeleton respectively. Figure 3.6 shows the graphs and extracted boundary and skeleton for a linear object. The sample points of the object are shown as black pixels beneath the two graphs in Figure 3.6.

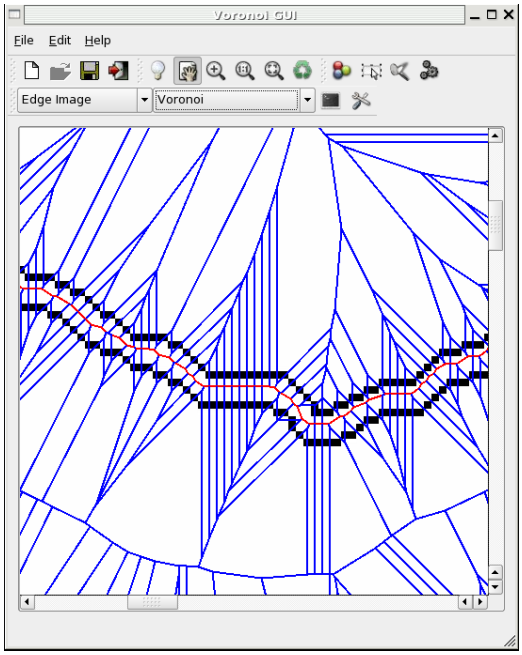
3.6 The Holistic Picture

The complete process can be generalized as shown in the flowchart in Figure 3.7. The primary user input during the process of skeletonization is the scanned map image. The user is also required to choose an object from the segmented image, but this can be made autonomous by processing all the objects in the image. An implementation of the procedure outlined in Figure 3.7 is done in the application VGUI. Some prominent features of this application are listed below.

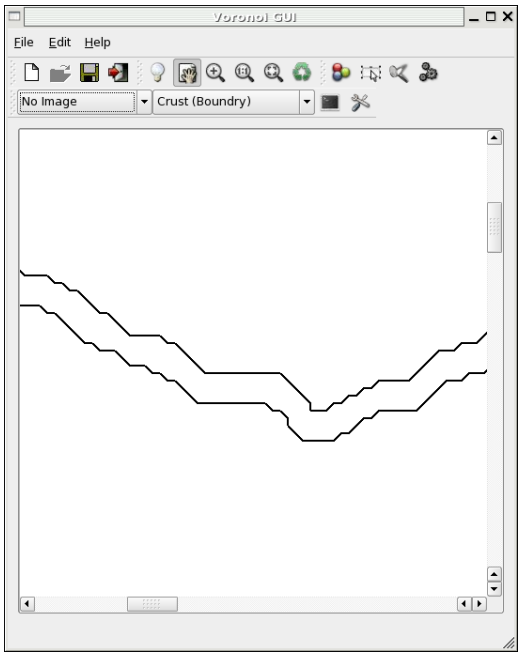
- It allows loading of images in various formats including JPEG, BMP, PNG and XPM.
- It allows interactive zooming and panning of the image.
- Regions can be added together to form a selection.
- It has a simple yet subjective interface for executing the skeletonization process.
- Various options to fine tune the process of skeletonization are provided via the *Settings* window.



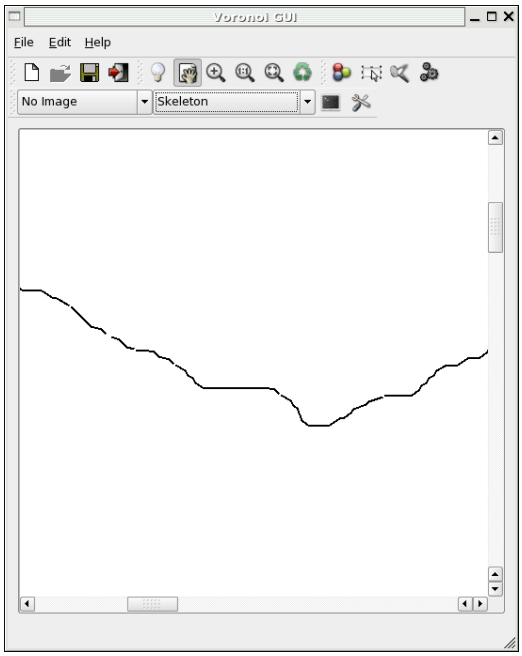
(a) The Delaunay triangulation



(b) The Voronoi diagram



(c) Extracted boundary



(d) Extracted skeleton

Figure 3.6: Skeleton extraction from colour image.

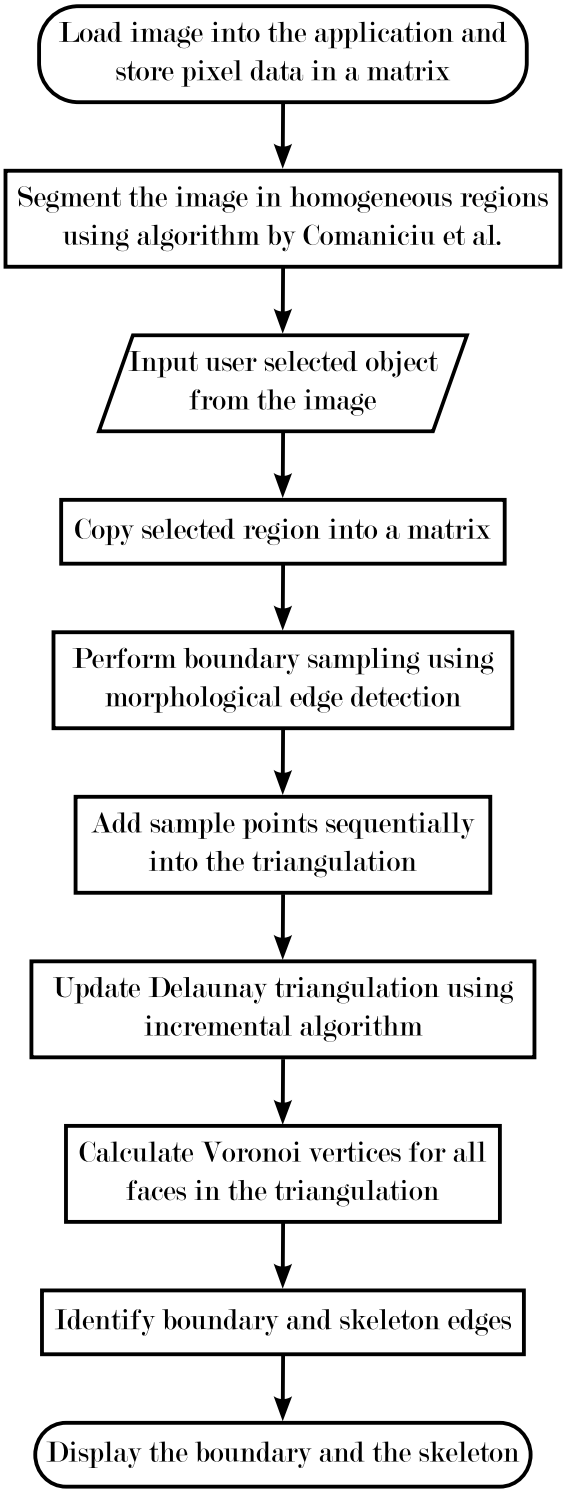


Figure 3.7: Skeletonization procedure

- Various image and vector graph overlay options provided to visualize and validate the results.

- Results can be saved as an image or exported as an ASCII file.

The application has a main interface window (see Figure 3.8) that gives access to the all the functionality. In order to process a new image, any previously opened image should be closed. Various display and other algorithm specific options are available through the settings window. Appropriate toolbar buttons are enabled and disabled as and when options for next possible steps come up. The *Canvas* can be zoomed, panned and refreshed at any time during the processing. Following steps highlight a typical workflow in the application VGUI. Refer to Figure 3.8 for the toolbar buttons.

1. **Open** an image by clicking the *Open image* button.

2. **Segment** the image by clicking on the *Segment image* button. Default segmentation mode is *quantization*. This can be changed from the *Settings* window.

3. **Select** the object of interest to digitize by activating the *Select object* button. As long as the *Select object* button is active, objects can be selected by clicking anywhere on the map inside the *Canvas*. Pressing *Shift + Click* will add objects into the selection, while pressing *Ctrl + Click* will remove an object from the selection. Panning is disabled during the selection but it can be toggled by pressing the *Select object* button.

4. **Sample** the object boundary by clicking the *Sample object boundary* button. This will detect edges around the selection and take all the points in the edge image as sample points.

5. **Extract** skeletons by pressing the *Extract skeletons* button. This will compute the Delaunay triangulation and the Voronoi Diagram followed by boundary and skeleton extraction. The skeleton needs to be pruned by clicking the *Prune leaf edges* button thrice. Every time this button is clicked, the leaf edges are removed from the skeleton.

6. **Display** the boundary and the skeletons by choosing one of the graphs from the *Vector graphs to overlay* drop-down list. A raster base image can be chosen from the *Base images* drop-down list.

7. The boundary or the skeletons can be either **Saved** as an image by pressing the *Save screen image* or exported as an ASCII file of line segments by pressing the *Export as ASCII* button.

8. A new image should be opened and processed only after **closing** the current image by pressing the *Close image* button.

The next chapter presents an examination of the pruning methods considered for skeletonization in this research.

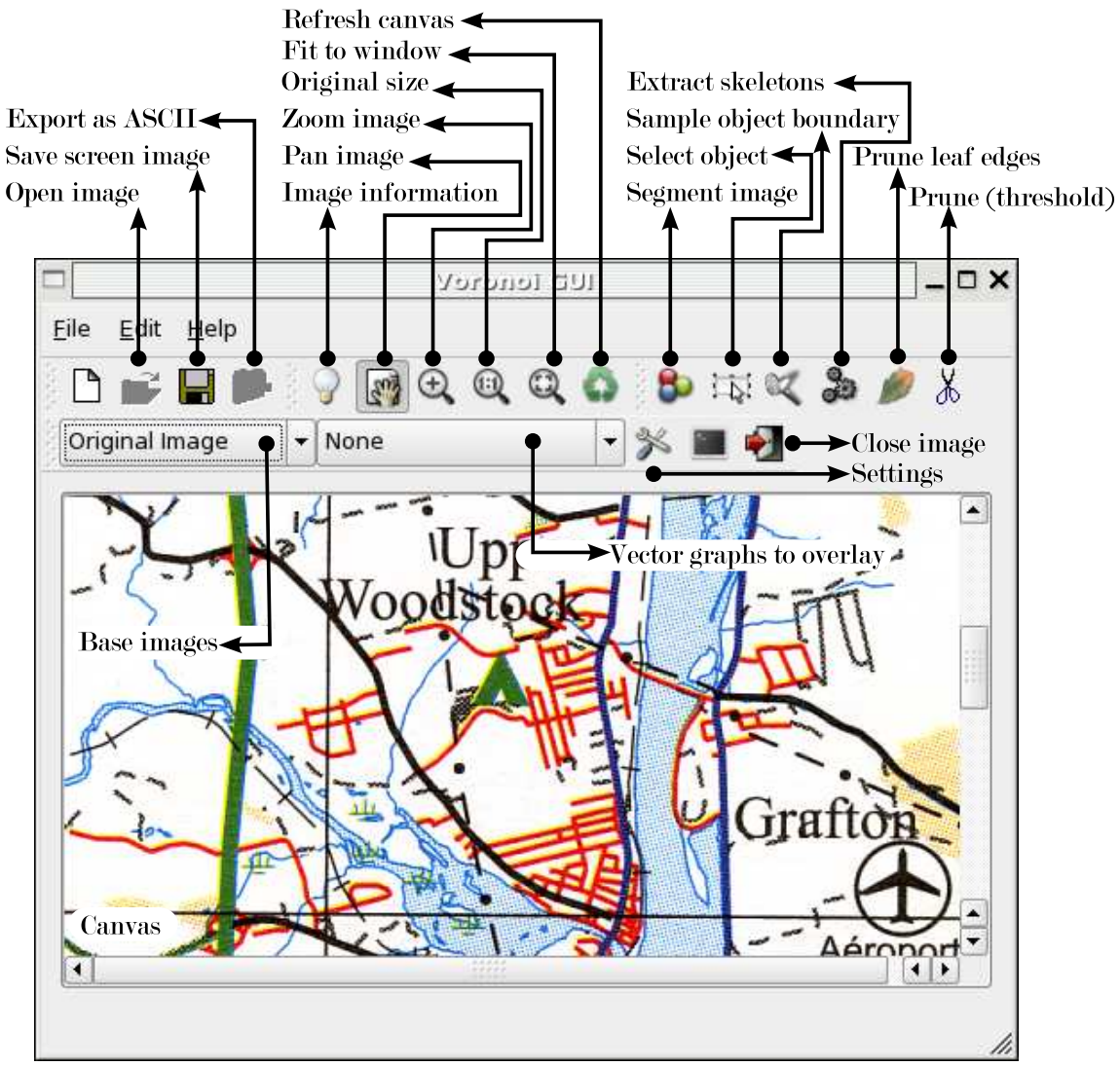


Figure 3.8: VGUI interface.

Chapter 4

Skeletonization

Chapter 3 discussed extension of the Delaunay/Voronoi graph based skeletonization algorithm to process colour images and implementation of image based object selection in an interactive application developed in this research. This chapter discusses a methodology to extract skeletons from the Voronoi diagram of a set of sample points around the selected object. This is followed by pruning in order to remove unwanted edges.

- Every quad-edge contains two Delaunay edges and two Voronoi edges (see Figure 4.1). Navigation from a Delaunay edge to a Voronoi edge is achieved by using the $Rot()$ operator and the reverse using the $Rot^{-1}()$ operator.

- The two Delaunay or Voronoi edges in a quad-edge are symmetric to each other as shown in Figure 4.1.

- Since the skeleton is a subset of the Voronoi diagram, only Voronoi edges will be considered as candidates for skeleton edges.

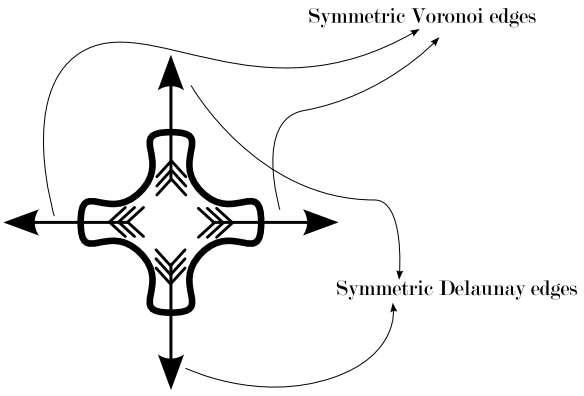


Figure 4.1: Four edges in the quad-edge structure.

4.1 Skeleton extraction from the Voronoi Diagram

Work of Amenta et al. [1998] shows that the "crust" or the boundary of a polygon can be extracted from an unstructured set of points provided the data points are well sampled. Gold and Snoeyink [2001] further simplify their method and show that the boundary can be extracted in a single step (see section 2.6). Gold [1999] discusses "anti-crust" in the context of skeleton extraction citing a brief introduction of this term in [Amenta et al., 1998]. The idea behind getting a skeleton is that a Voronoi edge is a part of the skeleton if its corresponding dual Delaunay edge is not part of the border set (crust) and it lies completely within the selected object. Thus, selecting the Voronoi edges lying inside the selected object that are the dual of the non-crust Delaunay edges should give us the skeleton (see Figure 4.2). The Voronoi edges thus selected form a tree structure called the "anti-crust" [Gold, 1999] that extend towards the boundary but do not cross it.

4.1.1 Obtaining Anti-crust from the Voronoi diagram

The anti-crust of an object, as described above, forms a tree-like structure that contains the skeleton. Once all the Delaunay edges belonging to the border set or

(a) The Voronoi diagram (b) Anti-crust (c) Skeleton

Figure 4.2: Skeleton abstracted from the anti-crust.

the crust are identified using the condition given by Gold [1999], it is easy to identify the Voronoi edges belonging to the anti-crust. In Figure 4.3, consider the Delaunay triangulation (dashed edges), the corresponding Voronoi diagram (dotted edges) and the crust edges (solid red edges).

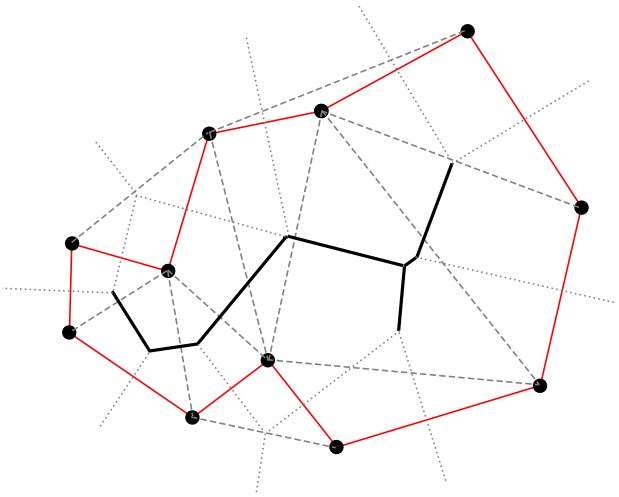


Figure 4.3: Anti-crust from the crust.

Navigation from a Delaunay edge to its dual Voronoi edge can be achieved by using the $Rot()$ operator in the quad-edge data structure. A Voronoi edge $e.Rot()$ of the dual Delaunay edge e is marked as an edge belonging to the anti-crust if the

following conditions are satisfied:

1. $e \notin Crust$
2. $e.Rot().Origin \in I$
3. $e.Rot().Destination \in I$

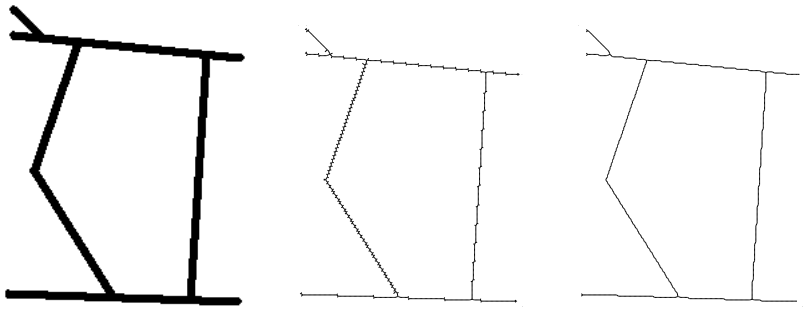
where $e.Rot().Origin$ is the origin coordinate of edge $e.Rot()$, $e.Rot().Destination$ is the destination coordinate of edge $e.Rot()$ and I is the selected object. This marks all the Voronoi edges belonging to the anti-crust that fall inside the selected object. Negating conditions (2) and (3) so that the coordinates do not fall inside the object will give us the exterior skeleton or the *exoskeleton*. Once the anti-crust is identified, an appropriate pruning method can be applied to get rid of the unwanted edges.

4.2 Pruning

Gold [1999] also discusses about the "hairs" around the skeleton that result due to the presence of three adjacent sample points whose circumcircle does not contain any other sample point - either near the end of a main skeleton branch or at locations on the boundary where there is minor perturbation because of raster sampling. Gold and Thibault [2001] suggest a skeleton retraction scheme in order to get rid of the hairs that also results in smoothing of the boundary of the object. Ogniewicz [1994] presents an elaborate skeleton pruning scheme based on various residual functions. Thus a hierarchical skeleton is created which is good for multiscale representation. Sharma et al. [2005] suggest the use of ratio based pruning in order to simplify a network of skeletons for extracting linear features from satellite imagery.

The problem of identifying skeleton edges now reduces how best to reasonably prune the anti-crust. Since we are concerned with centerlines of map objects that are

primarily linear features, a simple pruning method such as the ratio based pruning suggested by Sharma et al. [2005], pruning leaf edges or even a threshold based pruning will serve the purpose. Figure 4.4(b) shows the anti-crust of a linear feature. Figure 4.4(c) shows skeleton obtained from the anti-crust after pruning the leaf edges. Three methods considered in this research are threshold based pruning, pruning by Gold and Thibault [2001] with simplifications and ratio based pruning by Sharma et al. [2005].

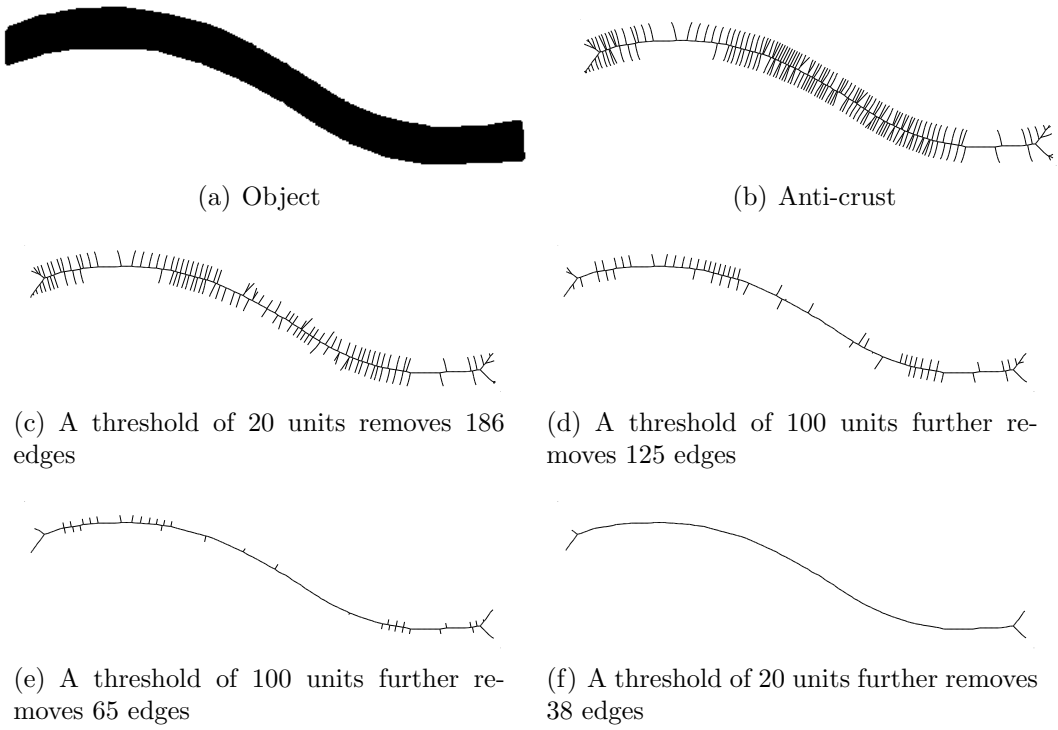


(a) Linear object (b) Anti-crust (c) Skeleton extracted by pruning leaf edges

Figure 4.4: Anti-crust of linear feature and extracted skeleton.

4.2.1 Threshold Based Pruning

In conventional GIS, pruning is generally achieved by specifying a threshold value. Any edge in the vector map is removed if its length is less than a threshold value specified by the user. This pruning method is considered in this research just in case the user is interested in manually removing some unwanted edges. Figure 4.5 shows the result of successive pruning by varying the threshold value. This method is suitable for interactive pruning but for automated pruning, other methods should be considered.



(a) Object

(b) Anti-crust

(c) A threshold of 20 units removes 186 edges

(d) A threshold of 100 units further removes 125 edges

(e) A threshold of 100 units further removes 65 edges

(f) A threshold of 20 units further removes 38 edges

Figure 4.5: Result of threshold based pruning.

4.2.2 Pruning by Removing Leaf Edges

Gold and Thibault [2001] present a retraction scheme for the leaf nodes in the anti-crust.The skeleton is simplified by retracting the leaf nodes of the skeleton to their parent nodes. Gold and Thibault [2001] recommend performing the retraction operation repeatedly until no further changes take place. An observation reveals that an unwanted branch in a skeleton may be composed of more than one edge (see Figure 4.6). Therefore, single retraction may not be sufficient to provide an acceptable skeleton.

A similar simplification can be achieved by pruning the leaf edges instead of retracting the leaf nodes. Leaf edge pruning produces satisfactory results and requires only two or three levels of pruning. Before pruning the leaf edges, these must be identified in the anti-crust using the operations provided by the quad-edge data

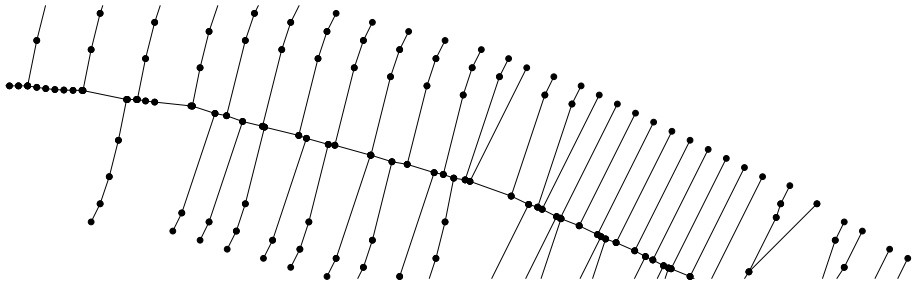


Figure 4.6: Hair around the skeleton composed of multiple edges.

structure. Figure 4.7 shows operators to access neighboring edges of an edge e.

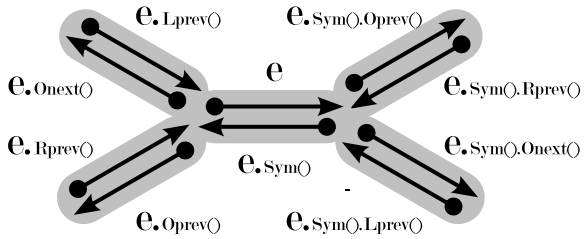


Figure 4.7: Accessing neighboring edges in a quad-edge.

An edge e from a tree of edges $T \in V$, where V is the Voronoi diagram, is marked as a leaf edge if the following condition is satisfied

$$\begin{gathered} e.Oprev() \notin T \text{ And } e.Onext() \notin T \\ \text{Or} \\ e.Sym().Oprev() \notin T \text{ And } e.Sym().Onext() \notin T \end{gathered} \tag{4.1}$$

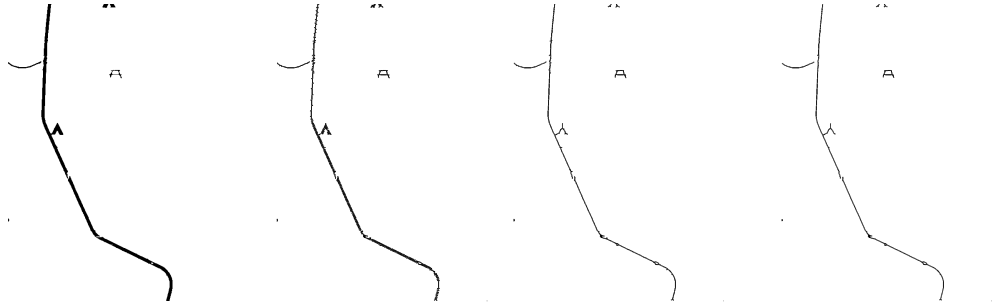
This condition essentially selects all the Voronoi edges belonging to the anti-crust that have at least one end point free (i.e., connected to an edge not belonging to the anti-crust). This condition is used to locate leaf edges followed by their removal from the skeleton. Experiments show that removing leaf edges two to three times simplifies the skeleton to a major extent for linear features.

Presented next are a few examples showing removal of "extraneous hair" from

the skeleton by pruning the leaf edges. Figure 4.8(a) shows a national highway selected from a scanned map. This serves as an example of a thick linear feature. The anti-crust of the highway is shown in Figure 4.8(b) with results of leaf edge pruning in Figures 4.8(c) and (d). A plot of the edges pruned in every iteration (see Figure 4.9) underlines the fact that the first two iterations are enough to produce an acceptable skeleton and that further pruning results in mere contraction of the skeleton (indicated by the horizontal plot after a steep drop after third iteration).

Another example shown in Figure 4.10 of roads selected from a scanned map serves as a case of thin linear features. Again the plot shown in Figure 4.11 confirms the adequacy of two iterations of pruning. It should be noted, however, that the case of thin linear features is even simpler than the previous case of thick linear objects due to the presence of a majority of single edge hairs. Since the plot shows a steep slope followed by a curve that is almost horizontal, one can expect most of the extraneous edges to be removed after the first iteration.

The last example shown in Figure 4.12 is a maple leaf that serves as an illustration of areal objects. In this research, areal features are not addressed by their skeletons but by their boundary, but it is interesting to see the effect of leaf edge pruning on such objects. The plot shown in Figure 4.13 shows a gradual decline in the number of edges pruned. This clearly indicates that more iterations are required to produce a good skeleton. In this example, the skeleton shown in Figure 4.12(f) was obtained after five iterations of pruning.



(a) National Highway (b) Anti-crust (c) Skeleton after pruning leaf edges (d) Skeleton after second pruning

Figure 4.8: Skeleton obtained after pruning leaf edges: Test image 1.

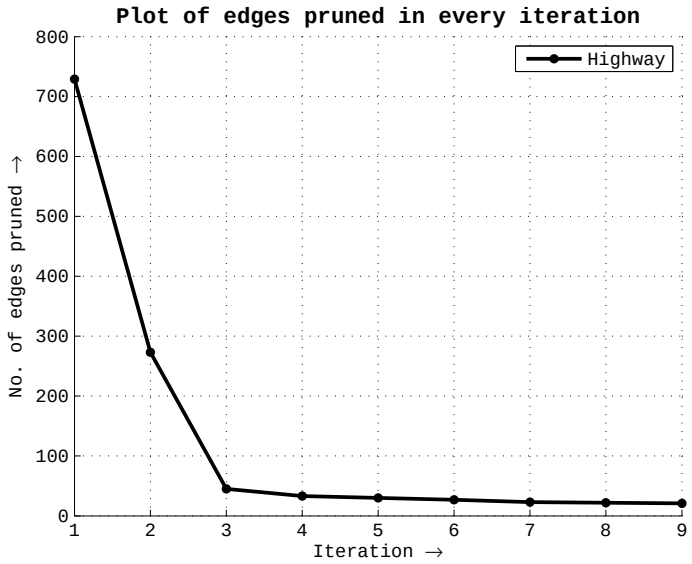
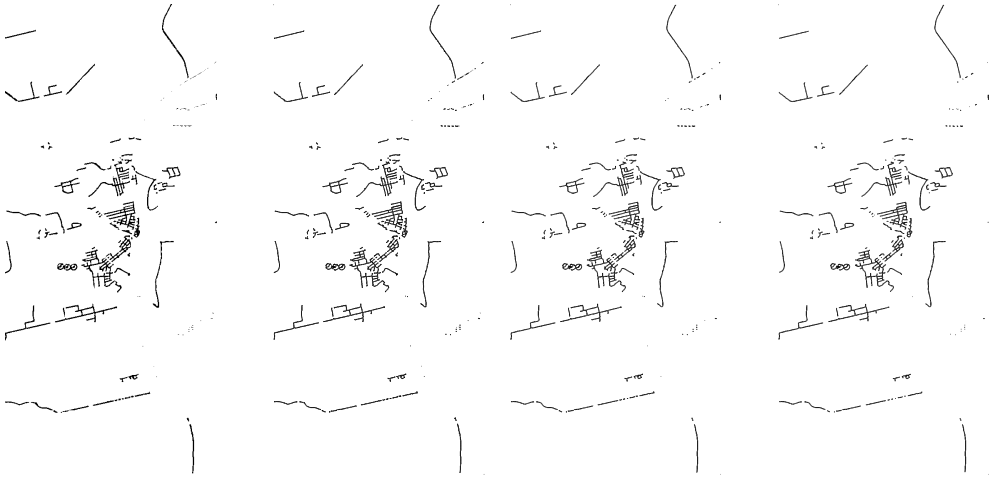


Figure 4.9: Pruning plot of object in test image 1.



(a) Roads (b) Anti-crust (c) Skeleton after pruning leaf edges (d) Skeleton after second pruning

Figure 4.10: Skeleton obtained after pruning leaf edges: Test image 2.

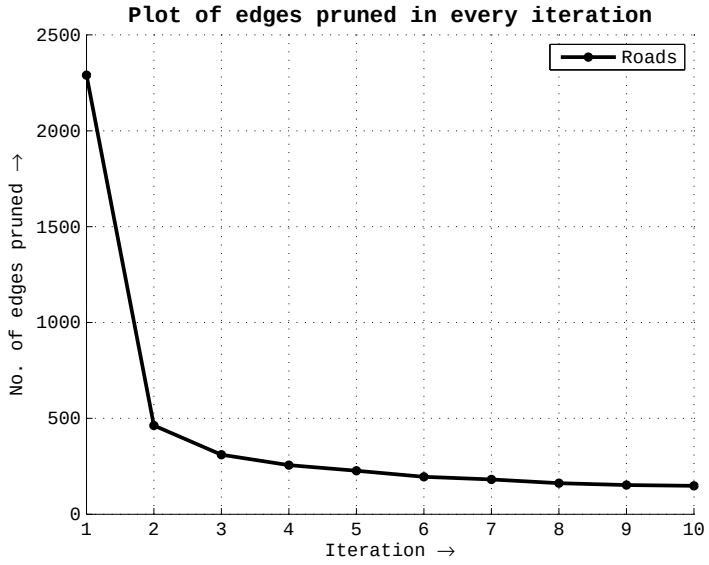
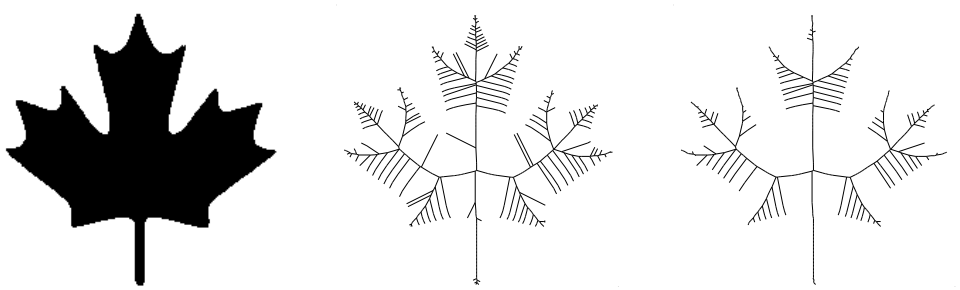


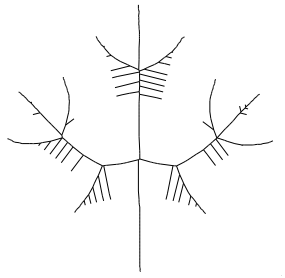
Figure 4.11: Pruning plot of object in test image 2.



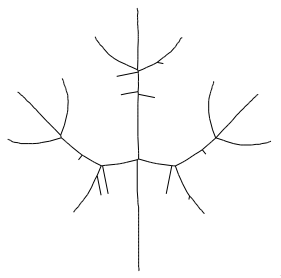
(a) Maple leaf

(b) Anti-crust

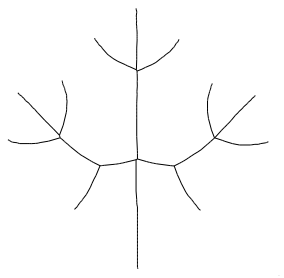
(c) Skeleton after removing leaf edges once



(d) Skeleton after removing leaf edges twice



(e) Skeleton after removing leaf edges thrice



(f) Skeleton after removing leaf edges five times

Figure 4.12: Skeleton obtained after pruning leaf edges: Test image 3.

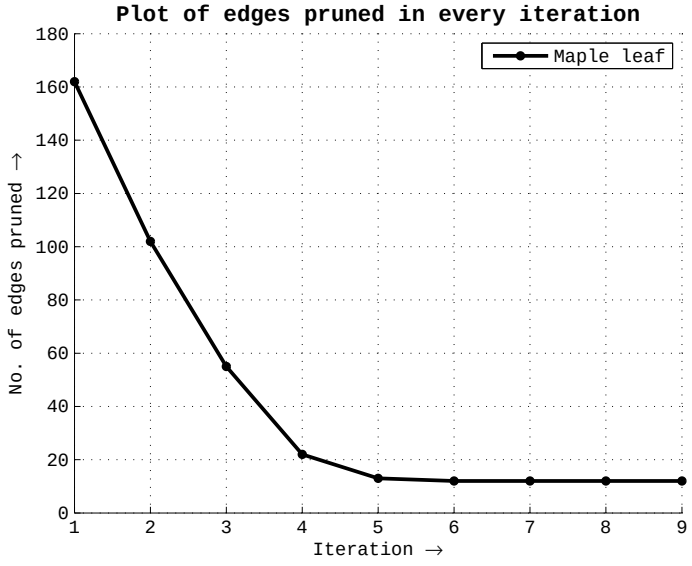


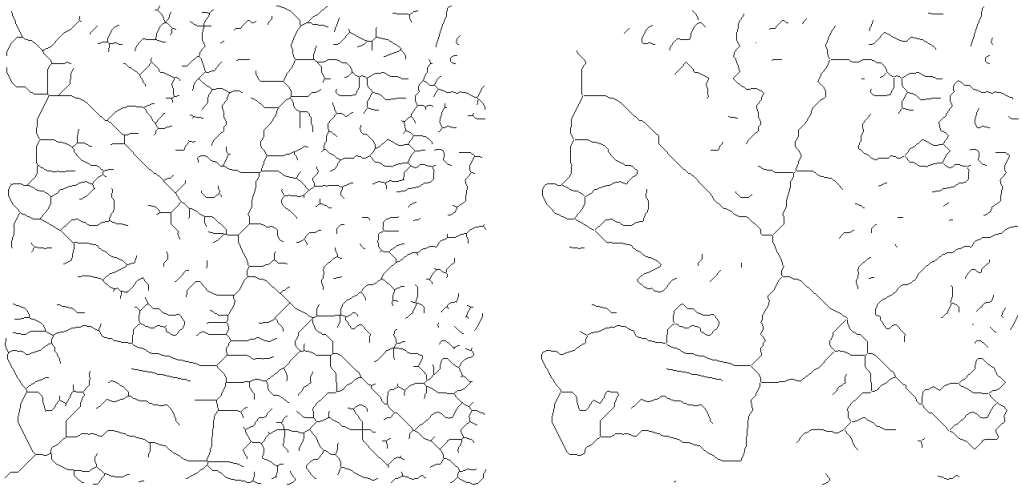
Figure 4.13: Pruning plot of object in test image 3.

4.2.3 Ratio Based Pruning of the Anti-crust

Work by Sharma et al. [2005] suggests a simple method of pruning with the intuition that the leaf edges having small lengths are more likely to belong to the clutter than the long edges. This method prunes all the leaf edges l_{leaf} that are less than a fraction f of the longest leaf edge L_{leaf}.

$$l_{leaf} \leq \frac{L_{leaf}}{f} \tag{4.2}$$

The value of the fraction f needs to be determined based on experiments. A value of $\frac{1}{2}$ is used by [Sharma et al., 2005]. This rule is applied iteratively on the network until no more edges are left to be pruned. Figure 4.15 shows the result of ratio based pruning applied on a skeleton.

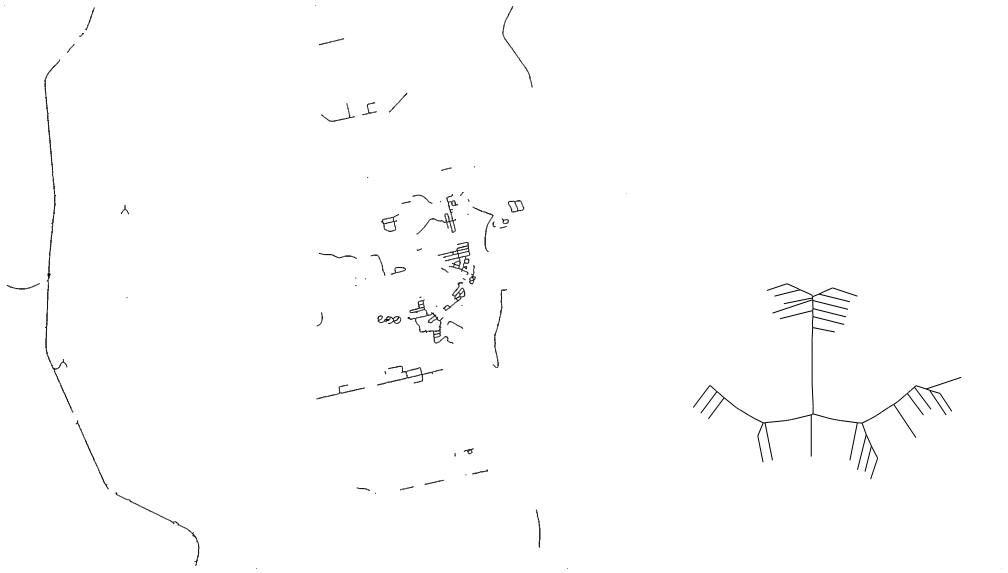


(a) Original skeleton (b) Pruned skeleton

Figure 4.14: Result of ratio based pruning.

Results of tests conducted with linear and areal objects to evaluate the ratio based pruning method are shown in Figure 4.15. The test results show that the pruning criteria works fine for linear features albeit with pruning that removes portions of the

main skeleton. For areal features, the results are not acceptable since the resulting skeleton still contains many spurious edges while parts of the main skeleton are pruned (see Figure 4.15(c)).



(a) Highway skeleton of object in Figure 4.12(a)

(b) Roads skeleton of object in Figure 4.12(a)

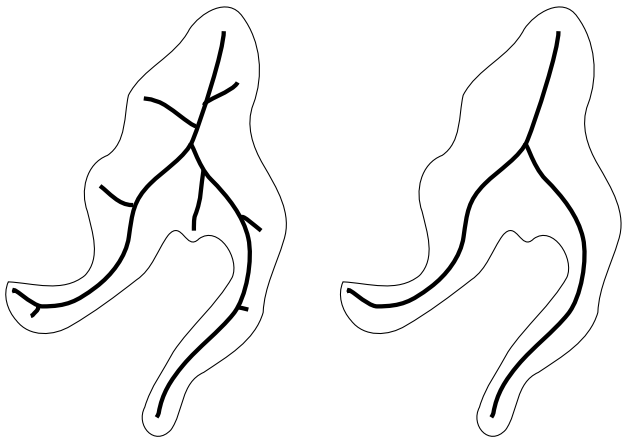
(c) Maple leaf skeleton of object in Figure 4.12(a)

Figure 4.15: Results of skeleton by the Ratio based pruning method.

4.3 Comparison of the pruning methods

Hadamard [1902] defined the concept of *ill-posed* problems in mathematics. An ill-posed problem is a problem whose solution either does not exist, is not unique or is not stable under changes in final data. Given a tree of edges, automatically pruning its edges (or branches) to obtain an acceptable skeleton seems to be an ill-posed problem. An argument in support of the above statement is the fact that perception of a skeleton differs from one individual to other. Figure 4.16 shows two perceptions of the skeleton of an object. One might argue that the perception depends on the

application but it is crucial to consider here that there is no concrete mathematical formulation for automated pruning. In other words, automated pruning is ill-posed because the solution is not unique and lack of any mathematical foundation behind preference of pruning an edge from a tree makes the task difficult.



(a) One possible skeleton (b) Another possible skeleton

Figure 4.16: Perception of a skeleton.

Following is a comparison of the pruning methods studied during this research based on the experiments that were performed on various maps.

Aspect(↓)	**Threshold based pruning**	**Leaf edge pruning**	**Ratio based pruning**
Can be automated?	No	Yes	Yes
Removes spurious edges efficiently?	Yes, depending on the threshold	Yes	Yes, for linear features
Leaves extraneous edges?	Depends on the threshold value	Minor	Yes

Therefore, leaf edge pruning is the method adopted in this research with two or three levels of pruning. The next chapter presents the results with their application.

Chapter 5

Results and Comparison

This chapter summarizes results of this research. The colour image segmentation algorithm discussed in Chapter 3 and the pruning method presented in Chapter 4 are tested. The skeletonization process needs minimal human intervention.

Extension of the existing skeletonization algorithm by Anton et al. [2001] to extract skeletons from the features on colour images has been one of the central objectives of this research. As explained earlier in Chapter 3, this is accomplished using a robust colour image segmentation algorithm developed by Comaniciu and Meer [1997]. Image segmentation allows partitioning the image into homogeneous regions based on their colour that has been achieved here by clustering using mean shift algorithm. The quality of the extracted objects, in terms of geometry and connectivity, depends on the quality of the scanned image. A noisy image may result in disconnected regions of an object that may produce broken skeleton. For sharp images with homogeneous regions, quantization produces good results while for noisy images, over-segmentation or under-segmentation (see section 2.4) gives satisfactory results. A low pass (blurring) filter applied to an image before processing it often results in a well segmented image since it subsides the high frequency textural details

and the noise (see Figure 5.1). The developed application gives the flexibility to the user to select and combine the regions. Thus, different regions of an object that were created after colour image segmentation can be combined together to form a single selection area.

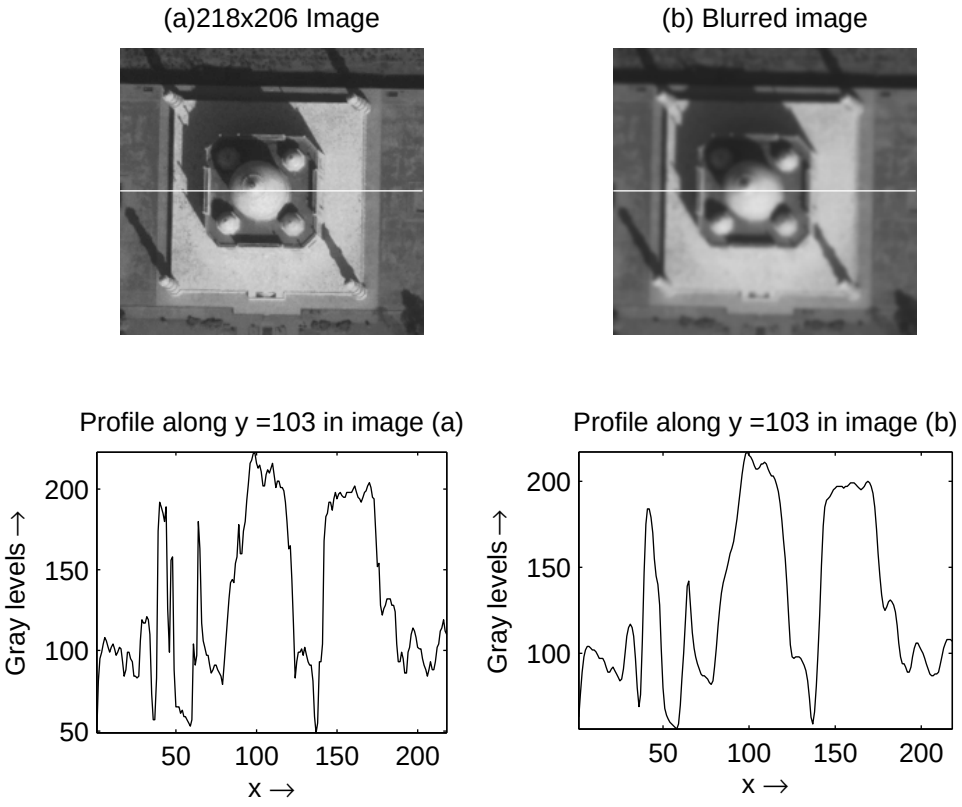
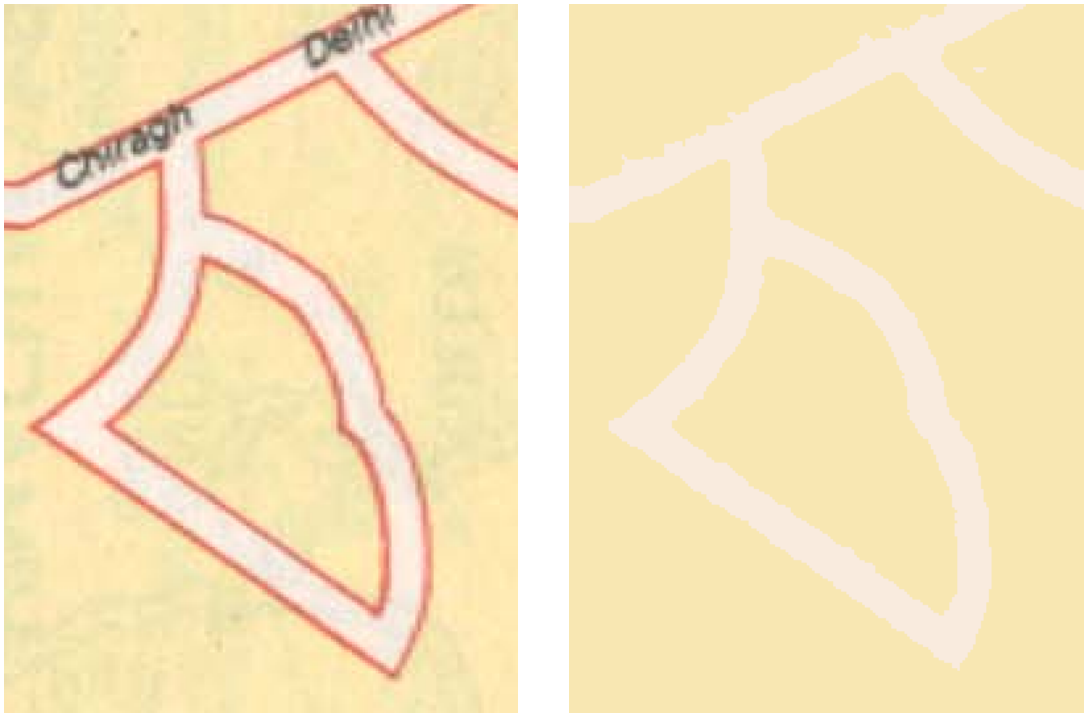


Figure 5.1: Effect of low pass filtering on image.

5.1 Feature Extraction Process

An example showing a complete map processing is presented here. The input image is a small portion of a scanned map (24 bit colour image) as shown in Figure 5.2(a). Since this image contains text labels and road boundaries, undersegmentation is used

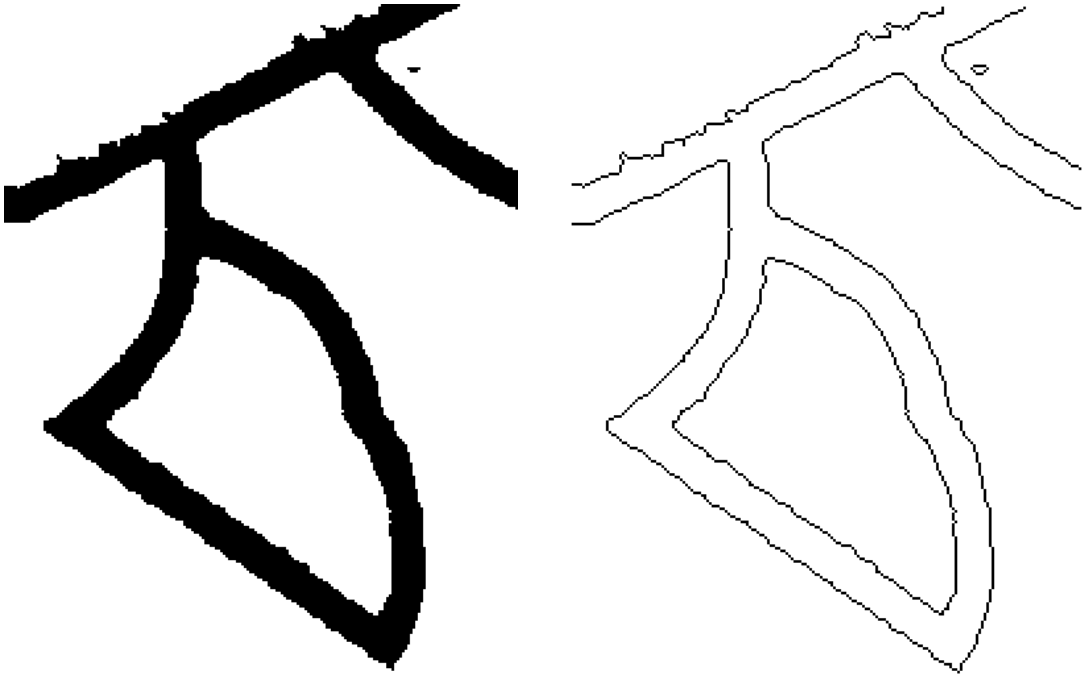
to remove these. This results in two colours in the segmented image as shown in Figure 5.2(b).



(a) Road map (b) Segmented image with two colours

Figure 5.2: Segmentation of scanned map.

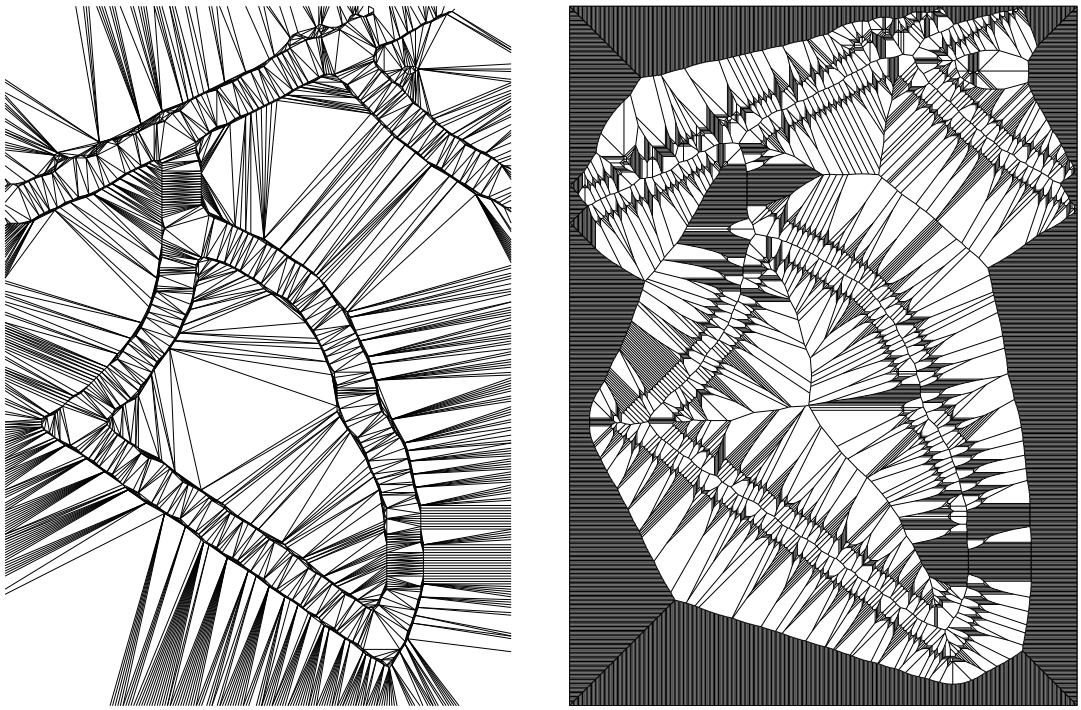
In the next step, the road object is chosen from the segmented image (see Figure 5.3(a)) and edge pixels are computed from the resulting binary image using morphological edge detection. The edge image is shown in Figure 5.3(b).



(a) Selected road (b) Edge pixels around the road

Figure 5.3: Object selection and boundary sampling.

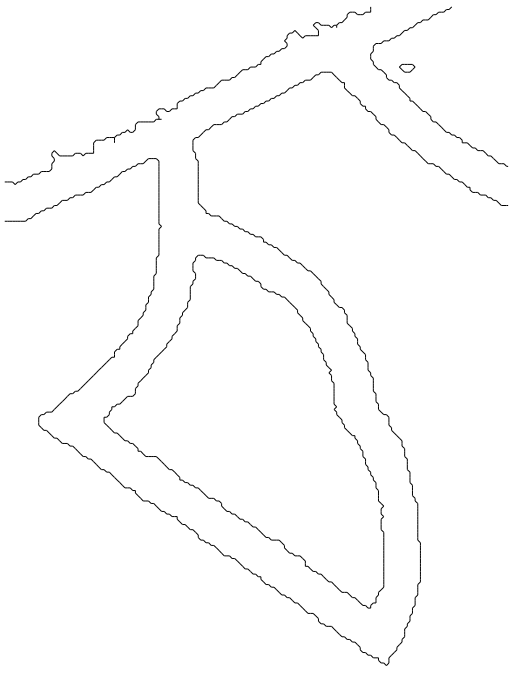
The sample points or the edge pixels are then used to compute the Delaunay triangulation (see Figure 5.4(a)) and the Voronoi diagram (see Figure 5.4(b)).



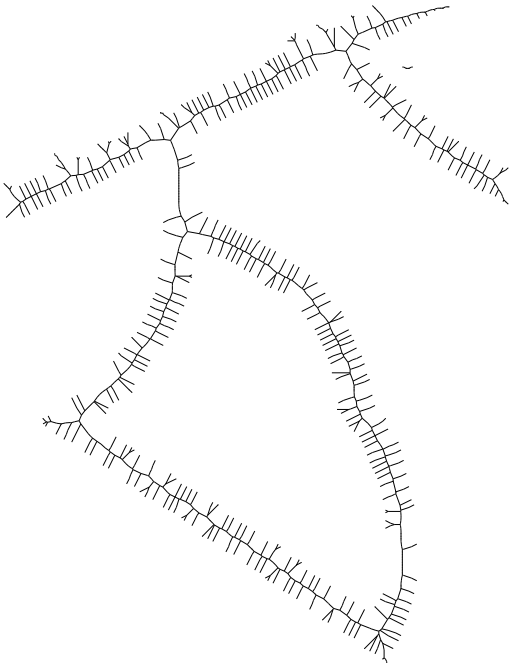
(a) Delaunay triangulation (b) Voronoi diagram

Figure 5.4: Delaunay triangulation and Voronoi diagram of the sample points.

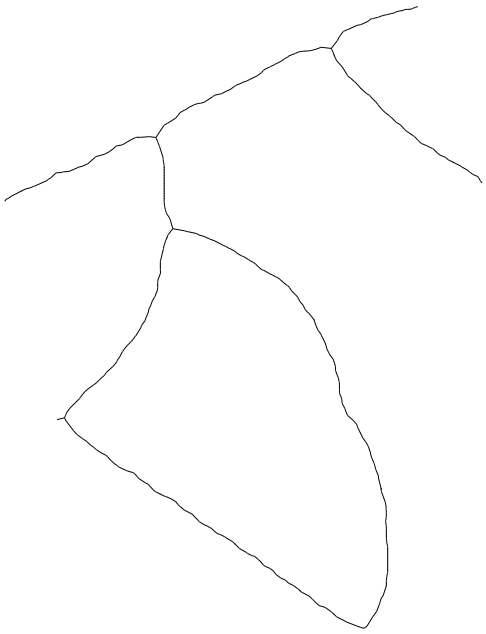
Using the crust extraction criteria, the boundary of the road object is extracted. This is shown in Figure 5.5(a). The anti-crust is then computed from the crust (shown in Figure 5.5(b)). After leaf edge pruning of the anti-crust, the skeleton is obtained (see Figure 5.5(c)).



(a) Crust computed from the Delaunay triangulation



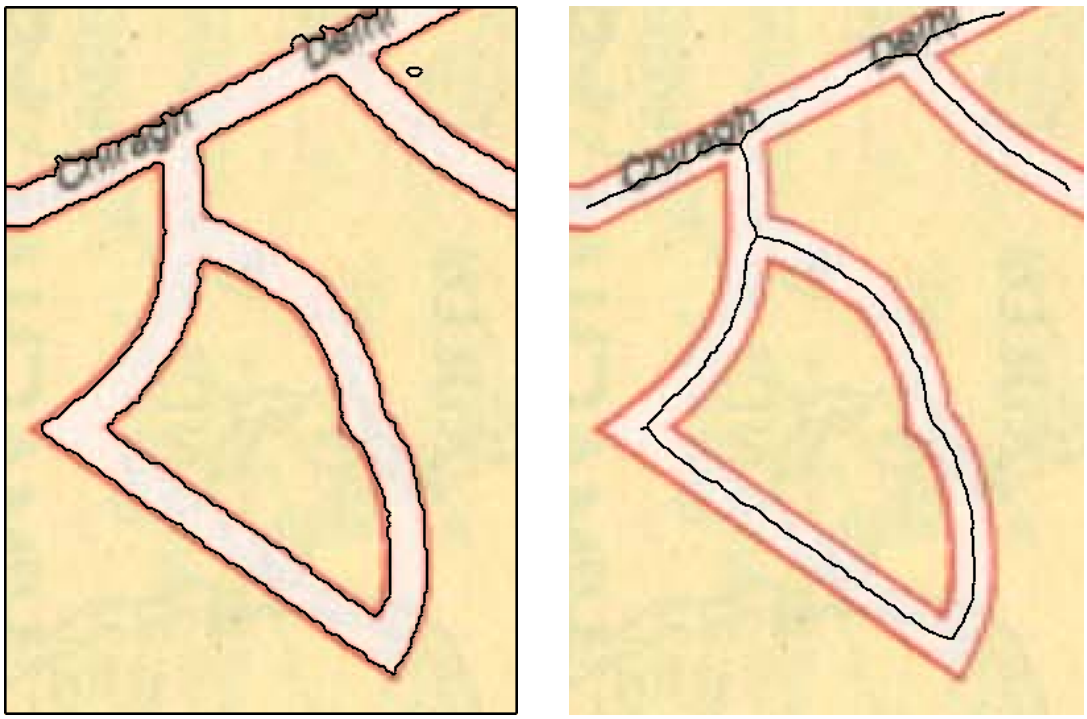
(b) Anti-crust computed from the Voronoi diagram



(c) Skeleton obtained from the anti-crust

Figure 5.5: Crust, anti-crust and skeleton of the object.

The extracted skeleton and the boundary are illustrated in Figure 5.6 by drawing them over the original map. Other results shown later in this chapter are obtained similarly.



(a) Extracted boundary (b) Extracted skeleton

Figure 5.6: Boundary and skeleton overlaid on the original scanned map.

5.2 Results with Scanned Maps

The skeletons and boundaries of map objects shown in this section are extracted using the algorithms implemented in the application VGUI. The skeletons are obtained by first computing the anti-crust and then pruning them by removing the leaf edges as discussed in Chapter 4. The boundary (crust) is generated using the single step algorithm [Gold, 1999]. A few examples of segmentation on scanned maps are shown next.

Figure 5.7(a) shows small portion of a map of the city of Moncton, New Brunswick, Canada [New Brunswick Atlas, 1998]. The map is scanned at 300 DPI (dots per inch) resolution and 24-bit colour depth. The map mainly consists of three types of roads: national highway (green), highway (blue) and streets (red). Crisp road edges with well defined road patches can be seen in the segmented image (Figure 5.7(b)) that are extracted as skeletons (see Figures 5.7(d), (f), and (h)). The skeletons show the presence of minor clutter due to misclassification during the segmentation process. Such a problem can be easily rectified in a GIS environment while manually editing the data after import.

The next example (see Figure 5.8) is a typical case of a dense urban road network. The extracted skeletons are of excellent quality and maintain good connectivity except at the places where other features were drawn over the roads in the original map. A knowledge driven approach is required to fill up the gaps arising due to missing information.

The next example shows a contour map obtained on-line from:

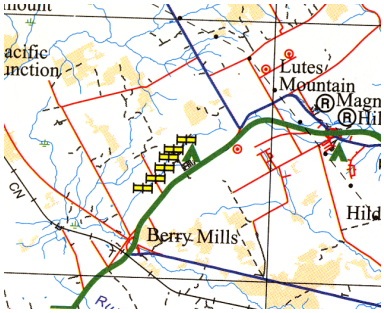
http://www.maptext.com/maps/Contour2.htm

Figure 5.9(b) shows a segmented image of the map. The map was treated with a low pass filter to even out noise in the image. The contour object in the segmented image is well separated from other objects (like roads and the background). Figure 5.9(d) shows the extracted centrelines of the contour curves, while Figure 5.9(f) shows the extracted road centrelines. The extracted contour centrelines, despite being broken at a few places, are excellent to be imported in a GIS, attach labels and optionally generate a digital elevation model.

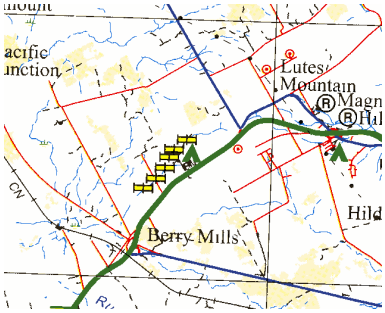
The map in Figure 5.10(a) is obtained from the proposal for Decision Support for Flood Event Prediction and Monitoring (FEPM) project set forth by the Emergency Measures Organization, New Brunswick, Canada. It shows flood risk mapping from

the 1973 flood in the Saint John river in Fredericton, New Brunswick and the risk of flooding in the area in a time span of 20 years (extracted in Figure 5.10(c) and (d)), 100 years (extracted in Figure 5.10(e) and (f)) and actual flood extent occurred in the year 1973 (extracted in Figure 5.10(g) and (h)). The flood plains are extracted as polygon vector boundaries from a crust of the selected object. Since the polygons were obtained after combining different regions, spurious boundaries are present inside the polygons. These boundaries should be removed during manual editing.

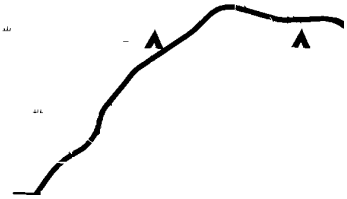
Figure 5.11(a) shows geographical area of interest for the same project. The area of interest was extracted as a skeleton (Figure 5.11(d)), while the Saint John river boundary extracted as crust (shown in Figure 5.11(f)). Some portions of the river polygon are missing because of the variation in river hue from blue to white that results in misclassification. This can be rectified either by introducing an intermediate step to flood-fill the selected region before computing the boundary or during manual editing in a GIS environment.



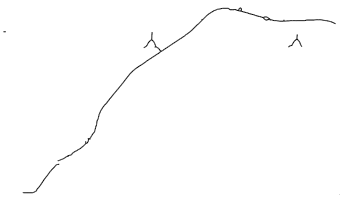
(a) Map with 62800 colours



(b) Segmented image with 10 colours

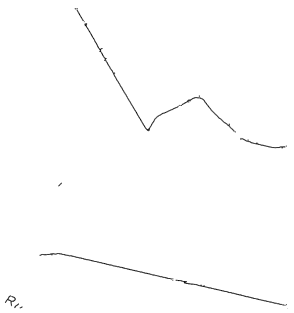


(c) National Highway

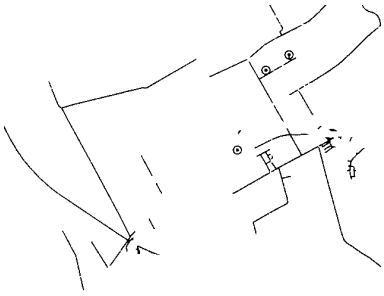


(d) Skeleton of national highway

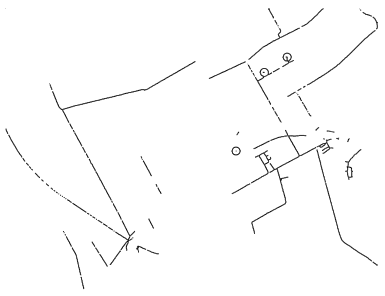
(e) Highway



(f) Skeleton of highway

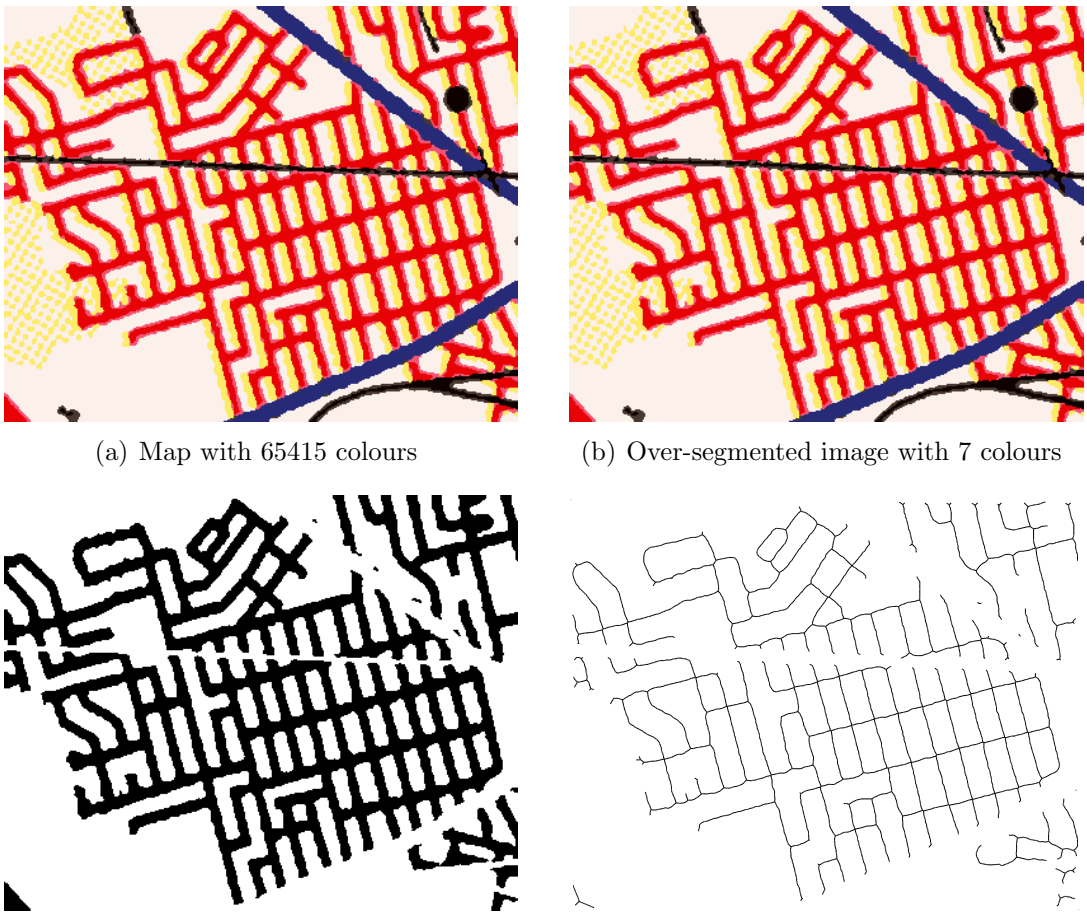


(g) Streets



(h) Skeleton of streets

Figure 5.7: Extraction of various roads from a map.



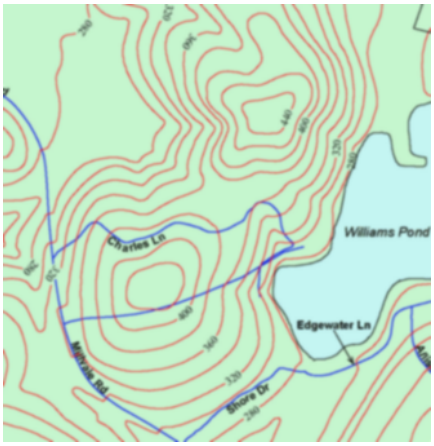
(a) Map with 65415 colours

(b) Over-segmented image with 7 colours

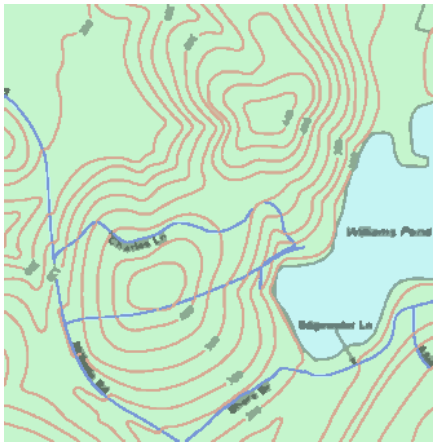
(c) Roads

(d) Skeleton of roads

Figure 5.8: Extraction of dense urban roads.

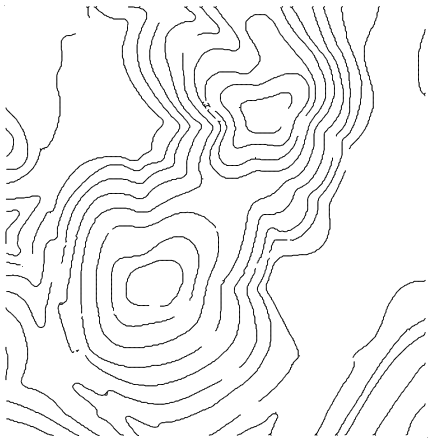


(a) Map with 8894 colours

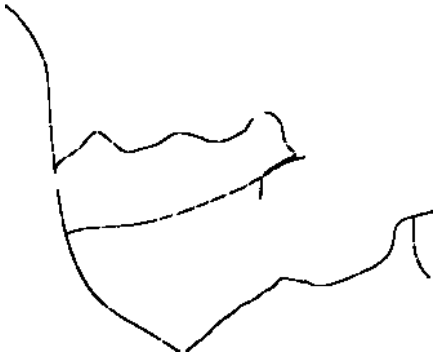


(b) Segmented image with 9 colours

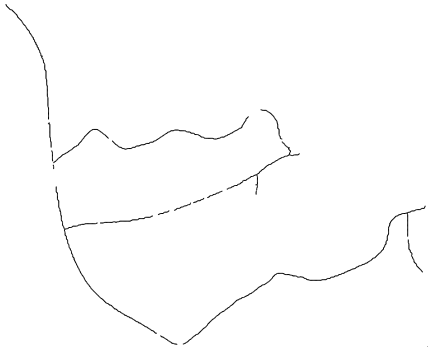
(c) Contours



(d) Skeleton of contours



(e) Roads



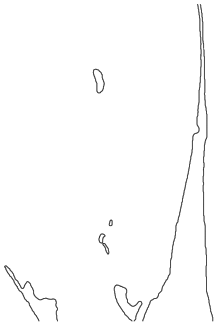
(f) Skeleton of roads

Figure 5.9: Extraction of contours and roads from a map.

(a) Flood extents in the city of Fredericton

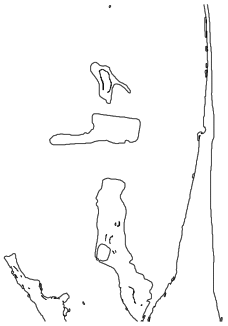
(b) Over segmented image 11 colours

(c) 20 years flood zone



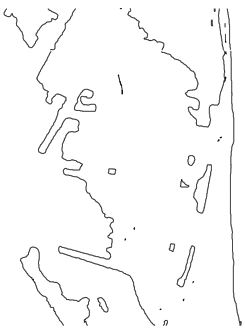
(d) extracted boundary

(e) 100 years flood zone



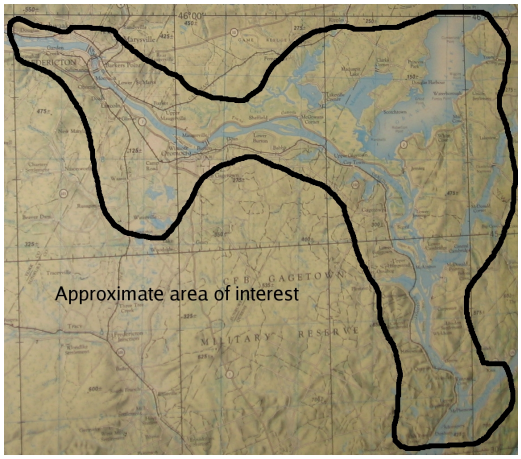
(f) extracted boundary

(g) 1973 flooded area

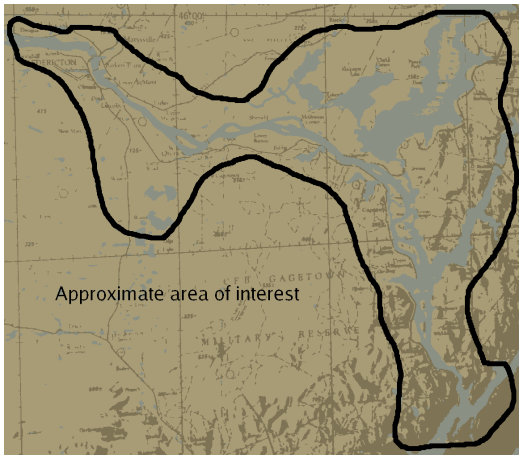


(h) extracted boundary

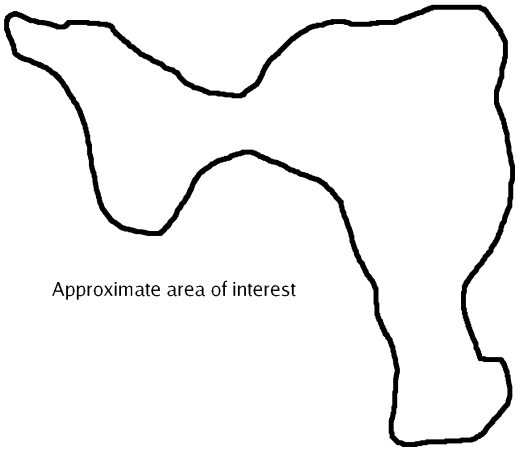
Figure 5.10: A portion of scanned flood risk map showing flood extents.



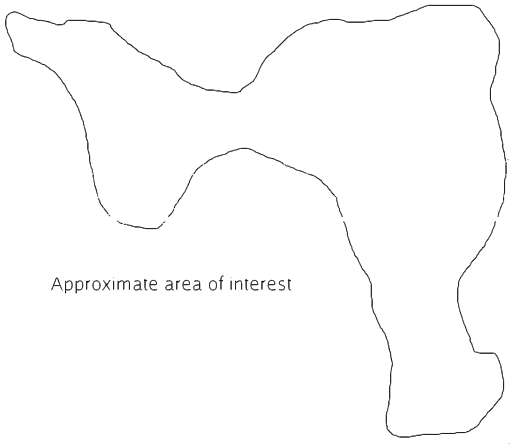
(a) Area



(b) Over segmented image

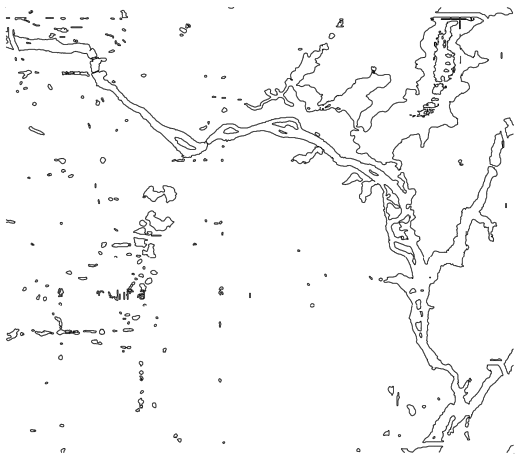


(c) Area of interest



(d) Skeleton of the area of interest

(e) Saint John River



(f) Extracted boundary

Figure 5.11: Geographical area of interest for the Flood Event Prediction and Monitoring (FEPM) project.

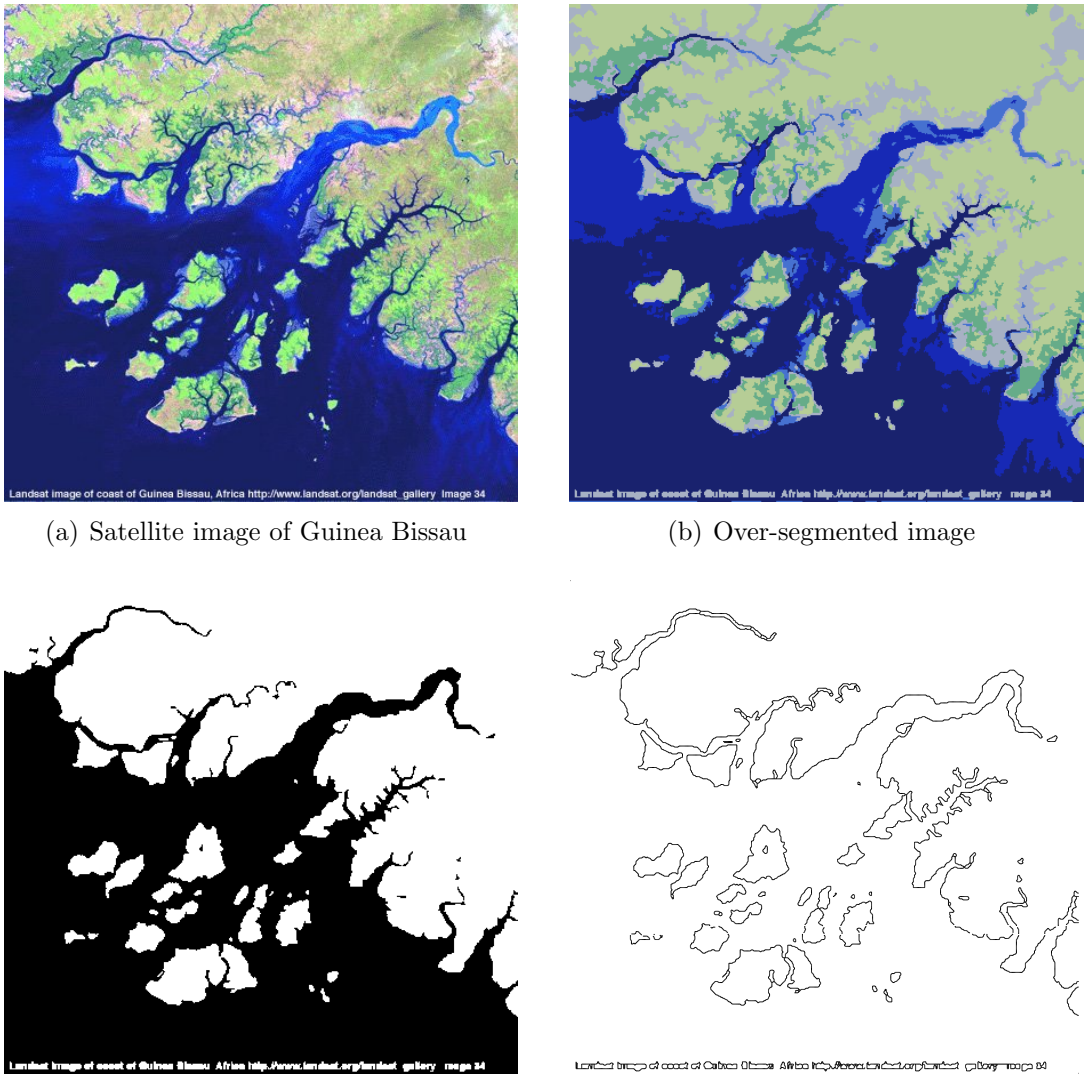
5.3 Results with Satellite Imagery

In order to test the applicability of the designed system to satellite images as well, a few experiments were conducted. Since the system is designed to make use of homogeneity in colors of objects, natural objects are better suited for our analysis. A few cases of coastline extraction have been considered here. A coastline is defined as the boundary between land and water. Coastline mapping is important for coastal activity monitoring, resource mapping, navigation, etc. A lot of work on coastline extraction from SAR (Synthetic Aperture Radar) and multi-spectral imagery has been done. Bo et al. [2001] provide a technique for coastline extraction from remotely sensed images using texture analysis. Liu and Jezek [2004] showed delineation of complete coastline of Antarctica using SAR imagery. Bagli and Soille [2003] suggest a morphological segmentation based automated approach for coastline extraction. Di et al. [2003] use the image segmentation algorithm by Comaniciu and Meer [2002] to segment an image and detect the shoreline. The final shoreline is obtained by a local refinement. In the following examples, the coastline is extracted as crust of the selected object. The accuracy of the coastline rendition depends on the spatial resolution of the imagery.

Figure 5.12 shows the complex coastline of Guinea Bissau. Segmentation results in four colours that define the water body out of which three define the coastline. Multiple selection enables combining these three regions together to form the complete coastline as shown in Figure 5.12(d).

Next, the coastline of Hebrides is shown in Figure 5.13. As can be seen, the coastline is corrugated. The extracted coastline is shown in Figure 5.13(d).

The beach of Seychelles shown in Figure 5.14(a) is mainly sandy and shows a wide variation in the ocean colour. The colour variation is primarily due to the depth



(a) Satellite image of Guinea Bissau

(b) Over-segmented image

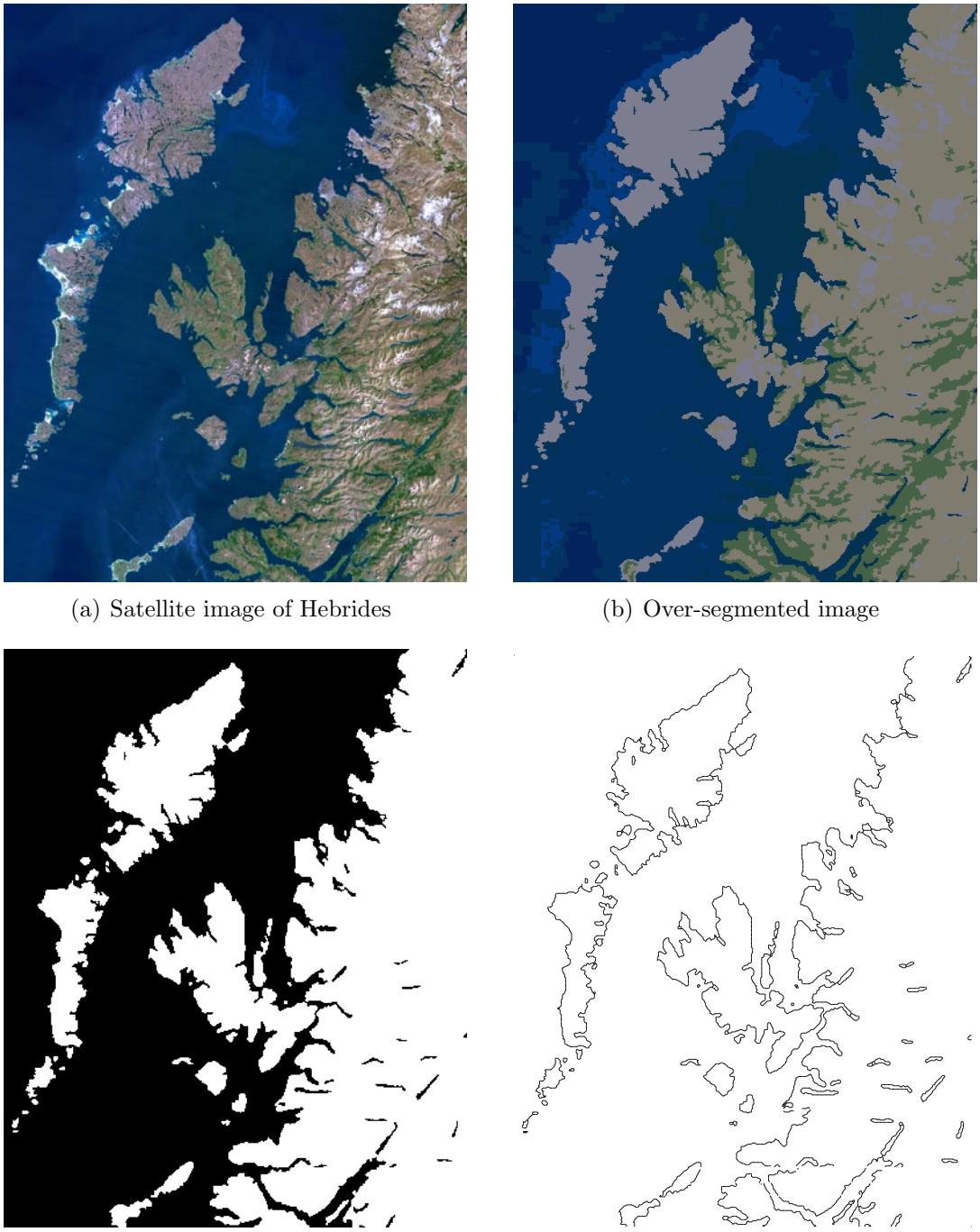
(c) North Atlantic ocean

(d) Extracted coastline

Figure 5.12: Feature polygon extraction from the satellite image of Guinea Bissau.

of water. Regions that form the ocean are combined to extract the coastline (see Figure 5.14(c)). The extended coastline is shown in Figure 5.14(d). The extended coastline shows the presence of a number of small polygons since the roads connecting the beach have the same colour value in the image and are included in the selection.

Satellite image of Lake Jempang, East Kalimantan, Indonesia (20 Jul 2002) ob-



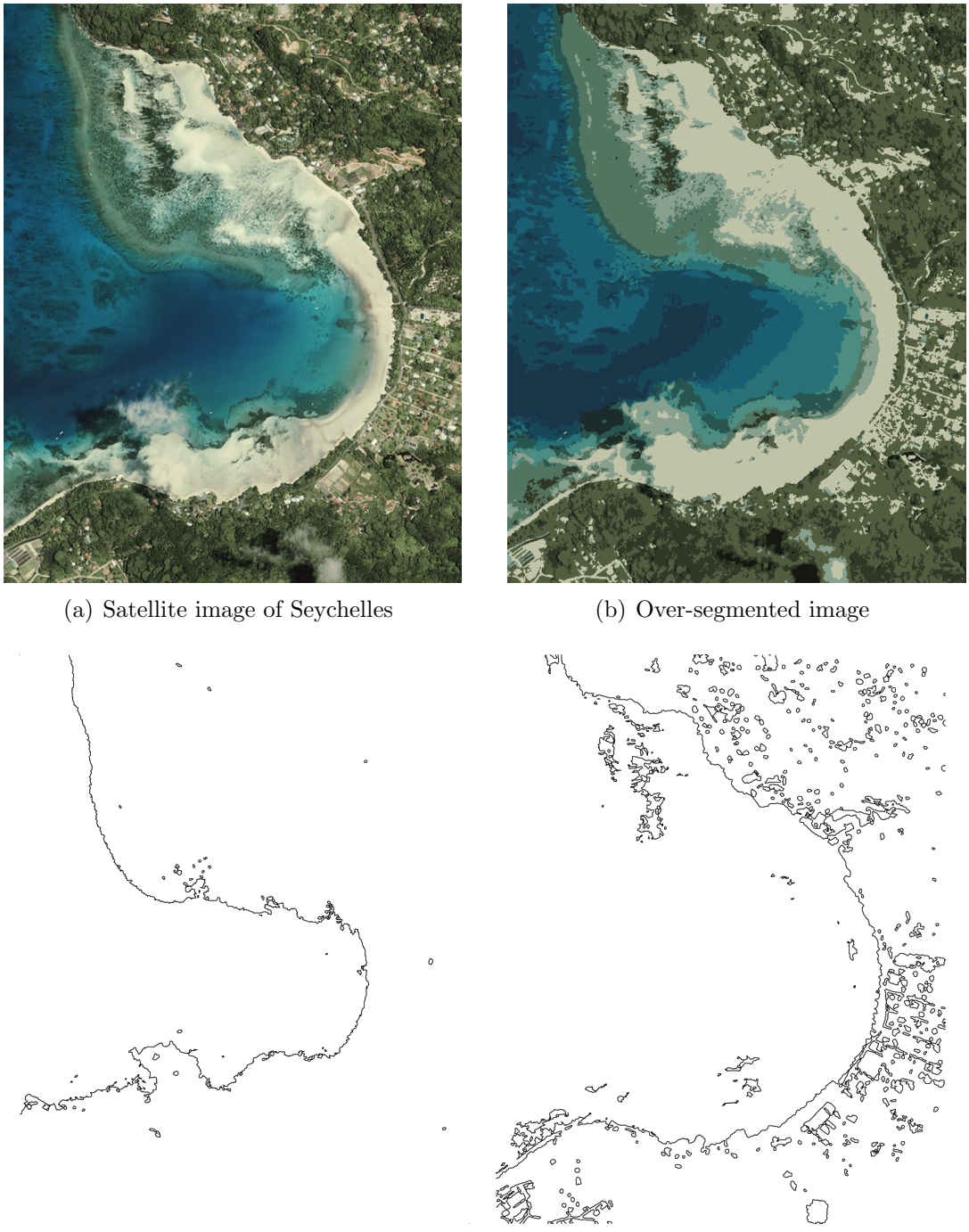
(a) Satellite image of Hebrides

(b) Over-segmented image

(c) Surrounding ocean

(d) Extracted coastline

Figure 5.13: Feature polygon extraction from the satellite image of Hebrides west coast.



(a) Satellite image of Seychelles

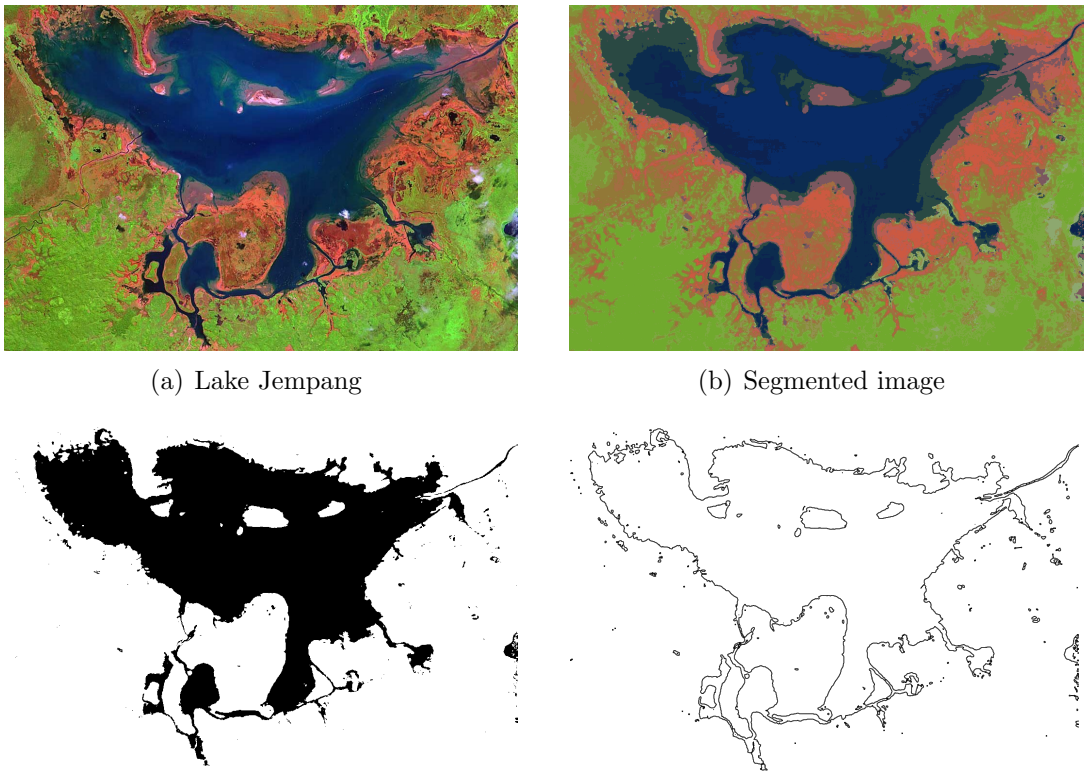
(b) Over-segmented image

(c) Extracted coastline

(d) Extended coastline

Figure 5.14: Feature polygon extraction from the satellite image of Seychelles.

tained online from Centre for Remote Imaging, Sensing and Processing (http://www.crisp.nus.edu.sg/monthly_scenes/y2002/Jul02_i.html) is shown in Figure 5.15(a). The extracted lake boundary is shown in Figure 5.15(d)).



(a) Lake Jempang (b) Segmented image

(c) Selected lake (d) Lake boundary

Figure 5.15: SPOT satellite image of Lake Jempang, East Kalimantan, Indonesia (20 Jul 2002).

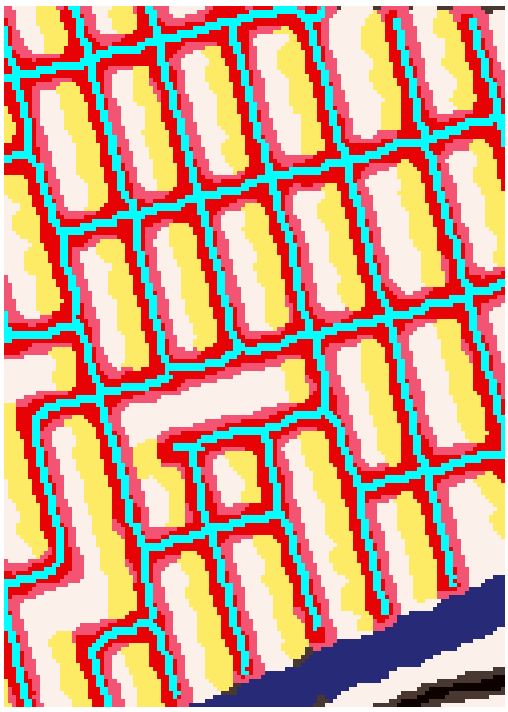
Applicability of the developed methodology can be easily extended to natural colour satellite imagery to extract homogeneous features. Coastline delineation, snow cover mapping, cloud detection, and dense forest mapping are a few areas where satisfactory results can be obtained. The next section gives a brief note about the positional accuracy of the skeleton and the boundary.

5.4 Positional Accuracy of Skeleton and Boundary

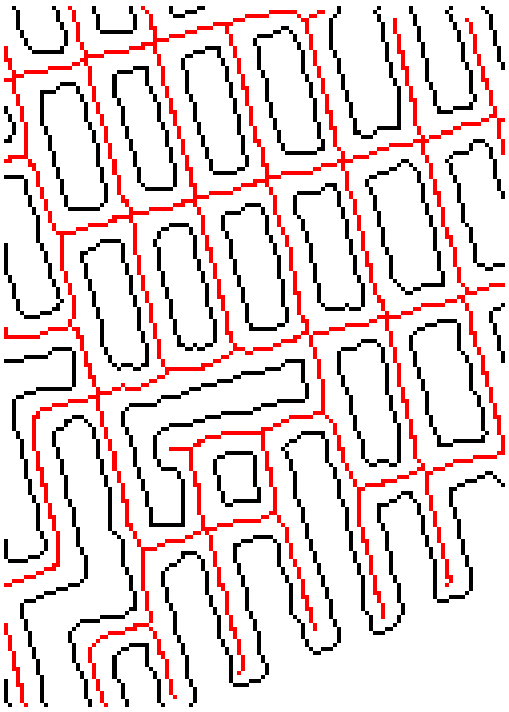
This section examines the positional accuracy of the extracted skeletons. Positional accuracy refers to how closely the extracted skeleton represents the actual object in the map. Here, we are mainly concerned with the condition that the skeleton falls within the object and lies approximately in the middle of it. This condition should be satisfied by objects of all thicknesses and geometries.

Presented next is an example to test this condition. A dense road network shown in Figure 5.16 highlights well-defined intersection points in the extracted skeleton. Figures 5.16(a) shows the skeleton obtained from the Voronoi diagram overlaid on the segmented image, while Figure 5.16(b) shows the same skeleton overlaid on the sample points. A comparison is required to analyse the accuracy of the skeletons obtained using the Voronoi diagram. Figure 5.16(c) shows the skeleton of the same road network using a morphological thinning method. An overlay of the skeletons from these two methods is shown in Figure 5.16(d) with the skeleton from Voronoi diagram in red and the one from morphological thinning in blue. This brings up two main points that the morphological thinning produces more than one pixel thick skeletons (at intersections) and removal of spikes needs extra processing. In general, the two skeletons match closely.

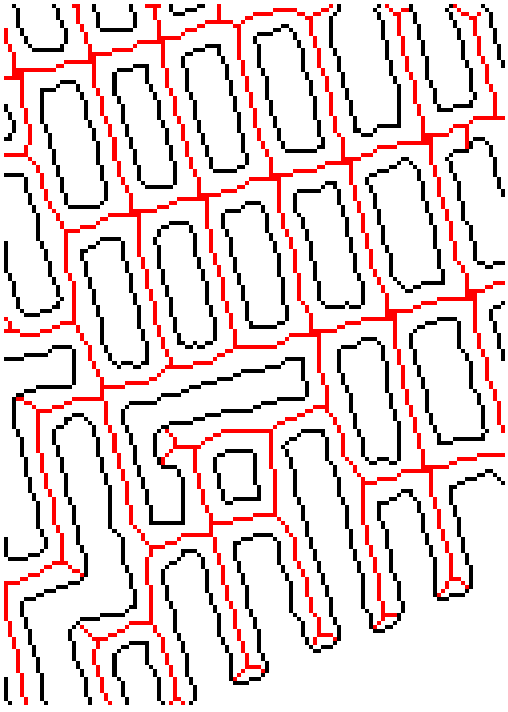
In order to verify the same condition for extracted boundaries as well, Figure 5.17 shows extracted polygon boundary of the Saint John river (Fredericton, New Brunswick) overlaid on a map. From Figure 5.17(b), it can be easily seen that the boundary follows the sample points closely.



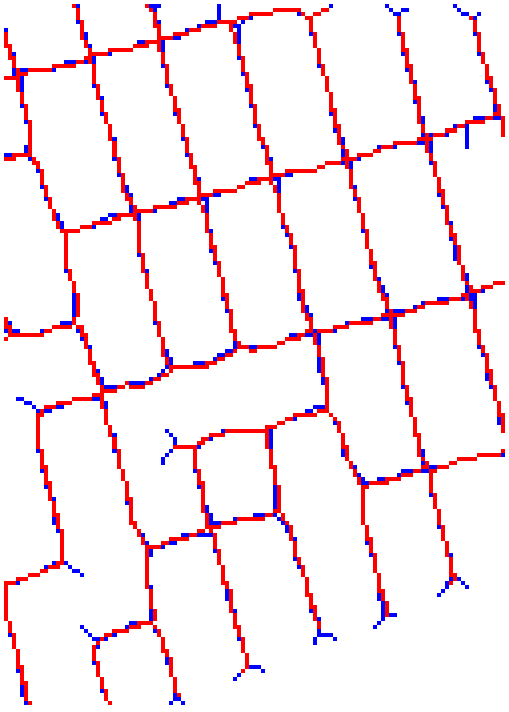
(a) Voronoi skeleton overlaid on map



(b) Voronoi skeleton overlaid on sample points

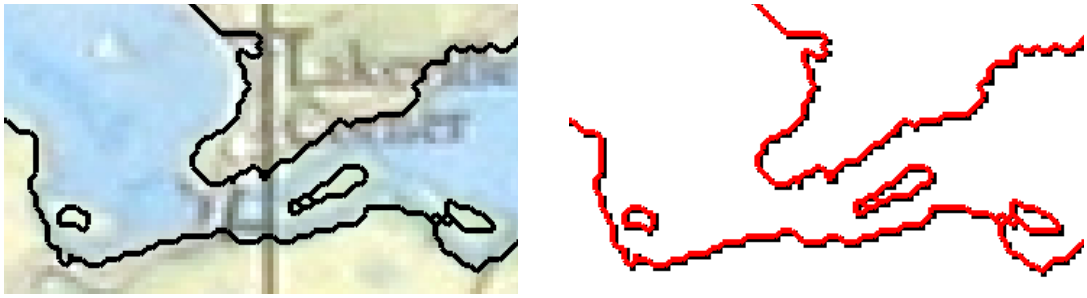


(c) Morphological skeleton overlaid on sample points



(d) Voronoi skeleton (red) overlaid on morphological skeleton (blue)

Figure 5.16: Positional details for skeleton of streets and comparison.



(a) Boundary overlaid on map (b) Boundary overlaid on sample points

Figure 5.17: Positional details for boundary of Saint John river.

5.5 Time Complexity

It is important to analyse the complete procedure for time complexity. Since the overall complexity will depend on the intermediate steps, these are first analysed individually.

Segmentation

Comaniciu [2000, p. 21] shows that the complexity of the probabilistic mean shift type algorithm that is employed in the segmentation algorithm [Comaniciu and Meer, 1997] is $O(mn)$, with $m \ll n$ where n is the number of pixels in the input image (or the number of feature vectors in the feature space) and m is the number of vectors in initial feature palette or clusters. Comaniciu [2000, p. 29] claims that the segmentation algorithm is linear with the number of pixels in the image.

Object selection

Object selection is implemented as a simple search of pixel colour values compared with the user selected points on the image. Therefore, it is $O(n)$ with the number of

pixels n in the image.

Edge detection

Binary edge detection is implemented as subtraction of two binary images. One of the binary images is always the original image while the other is either a dilated or an eroded image of the original. For an image I of size $x \times y$ and a structural element of size $k \times l$, computation of either erosion or dilation requires $x \times y$ iterations, each requiring $(k \times l) + 1$ comparisons. Therefore, the time complexity is linear with number of pixels of the input binary image and linear with the area of the structural element. Overall time complexity can be said to be $O(mn)$ where n is number of pixels in the image and m is number of pixels in the structuring element. Further, in the structural element, only locations having a value of 1 are considered in the computations. In our case, the application uses a 3×3 structural element with only five values being 1's in the mask (i.e., $m = 5$). Therefore, we can safely say that edge detection is $O(n)$ with the number of pixels n in the image.

Computation of the Delaunay triangulation and the Voronoi diagram

The quad-edge data structure allows computation of the Delaunay triangulation and the Voronoi diagram simultaneously. First the Delaunay triangulation is computed that builds the Voronoi diagram as well and then the Voronoi vertices are computed. The incremental algorithm to compute Delaunay triangulation updates an existing triangulation when a new point is inserted. This involves two basic operations: locating the triangle in which the new point falls and updating the triangulation (insertion). Locating a point in a triangle, using the "walking" described by Green

and Sibson [1977], terminates in $O(n)$ time for n vertices in the triangulation [Guibas and Stolfi, 1985, p. 121]. The total time for insertion of all n sites is $O(n^2)$.

Computation of border set

In the computation of the border set, the triangulation is traversed and an edge satisfying the criteria for the border edge is flagged as part of the boundary. This involves single traversal of the Delaunay graph and each edge traversal requires testing one condition. Therefore, border extraction is $O(n)$ with the number of edges n in the triangulation.

Computation of the Anti-crust

Computation of the anti-crust involves traversing the graph and identifying those Voronoi edges whose Delaunay edges are not a part of the border set. Furthermore, the edge should be completely inside the region selected by the user. This again gives rise to $O(n)$ complexity with the number of edges n in the triangulation.

Leaf edge pruning

Leaf edge pruning considers only those edges that belong to the anti-crust and tests each of them for being a leaf edge. The test completes with four comparisons. Therefore, the pruning step is $O(n)$ with the number of edges n in the anti-crust.

Therefore, all the steps except incremental computation of the Delaunay triangulation are linear with respect to their respective inputs. The whole procedure is a linear combination of these individual processes.

5.6 Step Ahead

This research shows a way to automatically obtain the centrelines by picking up objects from a scanned map. Even though there are many raster based methods for skeletonization (see section 2.1), Delaunay/Voronoi graph based skeletonization approach has an added advantage of providing *implicit topology* [Nonaka and Ohsawa, 2000]. Implicit topology defines object relationships within the data structure used to store it. There is no need to explicitly build it, as it is in the case of conventional GIS data structures. In our case, the quad-edge data structure efficiently defines the topology. GIS software like Dynamo, CARIS and spatial database systems like Oracle 10g support implicit topology. Despite the fact that the skeletons extracted here have implicit topology, they are incomplete due to the presence of gaps and clutter.

The skeletons shown here have been exported as an ASCII file from the application VGUI. The exported edges are then plotted in Matlab® and the plot is exported as a postscript image by running a script. In order to use the extracted skeletons and boundaries in a GIS environment, the following operations need to be performed:

1. Import the edges in a GIS environment.
2. Remove erroneous edges and cleaning.
3. Join disjoint curves and correcting wrong linkages.
4. Attach labels.

5.7 Advantages of this Research

The emphasis in this research has been to implement the skeletonization process for colour images using the Delaunay triangulation and the Voronoi diagram. The

standpoint on using Delaunay triangulation and Voronoi diagram for skeletonization is supported by the argument that they provide topology with the extracted skeletons while preserving the geometry. The emphasis here is on topology, which is a strong descriptor of any spatial relationship. The following are the main advantages and contributions of this research:

- It provides an effective framework for serializing the process of skeletonization as depicted in the flowchart in Figure 3.7. Utilizing colour segmentation for object selection followed by skeletonization is an efficient approach to the problem.
- Colour scanned maps can be processed appropriately.
- Leaf edge pruning provides excellent skeletons from the anti-crust obtained from the Voronoi diagram of the sampled points.
- Applicability of the methodology to satellite imagery has been shown by extracting natural features like coastlines.

5.8 Limitations of this Research

In general, Voronoi diagram based skeletonization methods are under-explored because of the complex nature of the extraction methods. Telea and van Wijk [2002] state:

"*The Voronoi diagram is the boundary's medial axis. Such methods produce an accurate connected skeleton, but are fairly complex to implement, require a robust boundary discretization, and are computationally expensive.*"

Lack of an existing autonomous pruning algorithm has been a major obstacle in comparing the obtained results. The following limitations have been observed with

this research work:

- Accuracy of the resulting skeletons can be verified quantitatively if the object has a mathematical representation. For objects in digital images, a skeleton can only be analysed visually.

- The segmentation algorithm [Comaniciu and Meer, 1997], as is the case with all segmentation algorithms, may result in misclassification depending on the image quality. Some image pre-processing is suggested for such cases.

Despite these limitations, this research presents an excellent methodology for raster to vector conversion. The way raster to vector conversion works in conventional GIS needs to be revised in order to integrate implicit topology during the conversion process. This will not only reduce errors in digitization but also make the final maps topologically correct.

Chapter 6

Conclusions and Future Work

This research work was started with three objectives in mind.

- Extend the existing Delaunay/Voronoi graph based skeletonization algorithm to colour images and improve it.
- Design an efficient and autonomous method for feature extraction from colour scanned maps.
- Develop an interactive application to streamline the process of skeletonization from scanned maps.

Although a complex research work like this needs more time to be completed, an effort has been made to achieve all of the objectives of this research and improve the existing work. In the given amount of time of about an year, a number of developments and findings were made in this research. This chapter concludes the research work done and provides some recommendations for improvements.

6.1 Conclusions

This research work succeeds in achieving its primary goals by designing an effective methodology for automated vectorization of features from scanned maps. The methodology enables extraction of centerline as well as boundaries of an object in a single step. The centerlines are required while digitizing linear features like roads, contours and streams. On the other hand, boundaries are needed to obtain polygons from features like fields and islands.

Based on the methodology, an interactive software application has been developed. It incorporates object extraction from input images using colour image segmentation. Segmentation based on clustering using a mean shift algorithm in feature space has been adopted here. The mean shift algorithm is a popular and robust method of clustering and provides good results in segmentation. The application allows selection of multiple objects for extraction of the skeleton or the boundary. Skeletonization and boundary extraction is based on the Voronoi diagram and the Delaunay triangulation. This not only gives accurate skeletons and boundaries, but also preserves topology of the extracted feature. It is worth noticing here that the extracted feature might not be a complete map object in the scanned image. This is due to labels and other features drawn over the feature of interest on a paper map.

An extensive study of commercial raster to vector conversion software packages has been done by Dharmaraj [2005]. The majority of these systems require black and white raster images with clearly demarcated objects. Some of the systems that process colour images provide inefficient object extraction which is based on thresholding and have a lack of topology in the extracted skeleton or boundary. The current methodology in this research tries to overcome these drawbacks. It gives excellent results with scanned maps. Experiments done with satellite imagery show acceptable

results too.

This research presents a prototype for automated digitization. The next process is to import the skeletons and the boundaries in a GIS environment. Cleaning the skeletons, joining gaps and labelling require additional work. Presented next are some recommendations on automating a few of the remaining tasks.

6.2 Future Work and Recommendations

There are certain key areas that either went unexplored or could not be researched well enough due to complexity of the research topic and time constraints. Following is a brief discussion of some recommendations.

Simplification of Extracted Features

The extracted features are in the form of series of line segments. These line segments are small since the sampling around the objects is very dense. This gives rise to huge amount of data. Furthermore, the general shape of the resulting polyline is jagged. These shortcomings can be overcome by simplifying the polyline using Douglas-Peucker curve simplification algorithm [Douglas and Peucker, 1973]. The skeleton can be smoothed by spline curve fitting.

Automated Gap Filling

Algorithms used in this research might generate discontinuous skeletons. Such gaps or discontinuities are a result of presence of unwanted features and text (labels) drawn over the feature of interest on a paper copy map. The following direction may be considered while designing an automated gap filling framework:

- Detection of small gaps should be studied considering not only the metric distance between the gaps but the direction of the tangent at the end points and proximity analysis. Once detected, gaps can be filled serially by joining the end points by parabolic or cubic curves keeping the tangents at the end-points intact (smooth transitions). This invariably requires an in-depth analysis of all the complex cases that can arise.

Detecting Text labels

Labelling a digitized feature is essential before it can be used for any sort of analysis in a GIS. Optical Character Recognition (OCR) systems can recognize text written on paper. Since OCR is still a new technology and is under development, the results may not be accurate. In case of scanned maps, a text recognition system can detect the label text and remove it from the map. This text can be later used for labelling the features under human supervision.

Bibliography

Amenta, N., M. Bern, and D. Eppstein (1998). "The crust and the β-skeleton: Combinatorial curve reconstruction." *Graphical models and image processing: GMIP*, Vol. 60, No. 2, pp. 125–135.

Anton, F., D. Mioc, and A. Fournier (2001). "2D image reconstruction using natural neighbour interpolation." *The Visual Computer*, Vol. 17, No. 3, pp. 134–146.

Bagli, S., and P. Soille (2003). "Morphological automatic extraction of coastline from pan-european landsat tm images." in *Proceedings of the Fifth International Symposium on GIS and Computer Cartography for Coastal Zone Management*, Vol. 3, Genova, pp. 58–59.

Bakker, W. H., and K. S. Schmidt (2002). "Hyperspectral edge filtering for measuring homogeneity of surface cover types." *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 56, No. 4, pp. 246–256.

Barber, C. B. (1993). *Computational geometry with imprecise data and arithmetic*, Ph.D. thesis, Princeton.

de Berg, M., M. van Kreveld, M. Overmars, and O. Schwarzkopf (2000). *Computational Geometry: Algorithms and Applications*, Springer, 2nd ed.

Bernard, T. M., and A. Manzanera (1999). "Improved low complexity fully parallel thinning algorithm." in *ICIAP '99: Proceedings of the 10th International Conference on Image Analysis and Processing*, IEEE Computer Society, Washington, DC, USA, pp. 215–220.

Bin, D., and W. K. Cheong (1998). "A system for automatic extraction of road network from maps." in *IEEE International Joint Symposia on Intelligence and Systems*, pp. 359–366.

Blum, H. (1967). "A transformation for extracting new descriptors of shape." in W. Wathen-Dunn, (Ed.) *Proceedings of the Symposium on Models for the Perception of Speech and Visual Form*, MIT press, Cambridge, MA, pp. 362–380.

Bo, G., S. Delleplane, and R. D. Laurentiis (2001). "Coastline extraction in remotely sensed images by means of texture features analysis." in *Geoscience and Remote Sensing Symposium, IGARSS '01*, Vol. 3, Sydney, NSW, Australia, pp. 1493–1495.

Bock, R. K. (1998). "Feature extraction." [On-line] 1 March 2006, http://rkb.home.cern.ch/rkb/AN16pp/node84.html.

Borgefors, G. (1984). "Distance transformations in arbitrary dimensions." *Computer Vision, Graphics, and Image Processing*, Vol. 27, No. 3, pp. 321–345.

Brown, K. Q. (1979). "Voronoi diagrams from convex hulls." *Information Processing Letters*, Vol. 9, No. 5, pp. 223–228.

Cheng, Y. (1995). "Mean shift, mode seeking, and clustering." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 8, pp. 790–799.

Comaniciu, D., and P. Meer (1997). "Robust analysis of feature spaces: color image segmentation." in *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, IEEE Computer Society, Washington, DC, USA, pp. 750–755.

Comaniciu, D., and P. Meer (2002). "Mean shift: A robust approach toward feature space analysis." *IEEE Transactions on Pattern Analysis Machine Intelligence*, Vol. 24, No. 5, pp. 603–619.

Comaniciu, D. I. (2000). *Non-Parametric Robust Methods for Computer Vision*, Ph.D. thesis, Rutgers, The State University of New Jersey.

Costa, L., and R. M. Cesar Jr. (2001). *Shape Analysis and Classification: Theory and Practice*, CRC Press.

Delaunay, B. (1934). "Sur la sphère vide." in *Bulletin of the Academy of the U.S.S.R., classe des Sciences Mathématiques et Naturelles*, 7(6), pp. 793–800.

Dharmaraj, G. (2005). *Algorithms for Automatic Vectorization of Scanned Maps*, Master's thesis, Department of Geomatics Engineering, University of Calgary, Calgary, Canada, http://www.geomatics.ucalgary.ca/links/GradTheses.html.

Di, K., J. Wang, R. Ma, and R. Li (2003). "Automatic shoreline extraction from high-resolution ikonos satellite imagery." in *Proceeding of ASPRS 2003 Annual Conference*, Vol. 3, Anchorage, Alaska.

Dony, R. D., and S. Wesolkowski (1999). "Edge detection on color images using rgb vector angle." in *Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, Edmonton, Canada, pp. 687–692.

Douglas, D., and T. Peucker (1973). "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature." *The Canadian Cartographer*, Vol. 10, No. 2, pp. 112–122.

Dwyer, R. A. (1987). "A faster divide-and-conquer algorithm for constructing delaunay triangulations." *Algorithmica*, Vol. 2, pp. 137–151.

Dwyer, R. A. (1989). "Higher-dimensional voronoi diagrams in linear expected time." in *SCG '89: Proceedings of the fifth annual symposium on Computational geometry*, ACM Press, New York, NY, USA, pp. 326–333.

Fisher, R., S. Perkins, A. Walker, and E. Wolfart (1996). *Hypermedia Image Processing Reference*, John Wiley and Sons.

Foley, J. D., A. van Dam, S. K. Feiner, and J. F. Hughes (1990). *Computer Graphics: Principles and Practice*, Addison-Wesley Publishing Company, 2nd ed.

Fortune, S. (1987). "A sweepline algorithm for voronoi diagrams." *Algorithmica*, Vol. 2, pp. 153–174.

Gabriel, K. R., and R. R. Sokal (1969). "A new statistical approach to geographic variation analysis." *Systematic Zoology*, Vol. 18, No. 3, pp. 259–278.

Gold, C. M. (1999). "Crust and anti-crust: A one-step boundary and skeleton extraction algorithm." in *Symposium on Computational Geometry*, ACM Press, New York, NY, USA, pp. 189–196.

Gold, C. M., and J. Snoeyink (2001). "A one-step crust and skeleton extraction algorithm." *Algorithmica*, Vol. 30, No. 2, pp. 144–163.

Gold, C. M., and D. Thibault (2001). "Map generalization by skeleton retraction." in *Proceedings of the 20th International Cartographic Conference (ICC)*, Beijing, China, pp. 2072–2081.

Gonzalez, R. C., and R. E. Woods (2002). *Digital Image Procesisng*, Prentice Hall, 2nd ed.

Green, P., and R. Sibson (1977). "Computing dirichlet tessellations in the plane." *The Computer Journal*, Vol. 21, No. 2, pp. 168–173.

Guibas, L., and J. Stolfi (1985). "Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams." *ACM Transactions on Graphics*, Vol. 4, No. 2, pp. 74–123.

Guibas, L. J., D. E. Knuth, and M. Sharir (1990). "Randomized incremental construction of delaunay and voronoi diagrams." in *ICALP '90: Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, Springer-Verlag, London, UK, pp. 414–431.

Guibas, L. J., D. E. Knuth, and M. Sharir (1992). "Randomized incremental construction of delaunay and voronoi diagrams." *Algorithmica*, Vol. 7, pp. 381–413.

Habib, A., Y. Lee, and M. Morgan (2003). "Automatic matching and three-dimensional reconstruction of free-form linear features from stereo images." *Journal of Photogrammetric Engineering and Remote Sensing*, Vol. 69, No. 2, pp. 189–197.

Hadamard, J. (1902). "Sur les problèmes aux dérivées partielles et leur signification physique." *Princeton University Bulletin*, Vol. 13, pp. 49–52.

Horowitz, S. L., and T. Pavlidis (1974). "Picture segmentation by a directed split-and-merge procedure." in *Proceedings of the 2nd International Joint Conference on Pattern Recognition*, Copenhagen, Denmark, pp. 424–433.

Huskey, P. (2002). "Electro magnetic spectrum." [On-line] 06 May 2006, http://newark.rutgers.edu/∼huskey/images/em_radiation.jpg.

Jähne, B. (2004). *Practical Handbook on Image Processing for Scientific and Technical Applications*, CRC Press, 2nd ed.

Kasturi, R., R. Fernandez, M. L. Amlani, and W. chun Feng (1989). "Map data processing in geographic information systems." *Computer*, Vol. 22, No. 12, pp. 10–21.

Kelley, J. L. (1955). *General Topology*, Van Nostrand Reinhold, New York, NY.

Kong, T. Y., and A. Rosenfeld (1989). "Digital topology: introduction and survey." *Computer Vision, Graphics, and Image Processing*, Vol. 48, No. 3, pp. 357–393.

Krämer, J. (1995). *Delaunay triangulation in two and three dimensions*, Master's thesis, Universität Tübingen, Institut für Informatik, Tübingen, Germany, http://www.gris.uni-tuebingen.de/gris/proj/dt/dteng.html.

Lee, K. H., S. B. Cho, and Y. C. Choy (1998). "A knowledge-based automated vectorizing system for geographic information system." in *ICPR '98: Proceedings of the 14th International Conference on Pattern Recognition - Volume 2*, IEEE Computer Society, Washington, DC, USA, p. 1546.

Liu, H., and K. C. Jezek (2004). "A complete high-resolution coastline of antarctica extracted from orthorectified radarsat sar imagery." *Photogrammetric Engineering and Remote Sensing*, Vol. 70, No. 5, pp. 605–616.

Maus, A. (1984). "Delaunay triangulation and the convex hull of n points in expected linear time." *BIT*, Vol. 24, p. 151163.

New Brunswick Atlas (1998). "Moncton." Map No. 21I/02, New Brunswick, Canada.

Niblack, C. W., P. B. Gibbons, and D. W. Capson (1992). "Generating skeletons and centerlines from the distance transform." *CVGIP: Graphical Models Image Processing*, Vol. 54, No. 5, pp. 420–437.

Nonaka, H., and Y. Ohsawa (2000). "An implicit topology description." in *IAPRS*, Vol. XXXIII, Amsterdam.

Ogniewicz, R. (1995). "Automatic medial axis pruning by mapping characteristics of boundaries evolving under the euclidean geometric heat flow onto Voronoi skeletons." Tech. Rep. 95-4, Harvard Robotics Laboratory.

Ogniewicz, R. L. (1994). "Skeleton-space: A multiscale shape description combining region and boundary information." in *Proceedings of Computer Vision and Pattern Recognition, 1994*, pp. 746–751.

Ogniewicz, R. L., and O. Kübler (1995). "Hierarchic Voronoi skeletons." *Pattern Recognition*, Vol. 28, No. 3, pp. 343–359.

Okabe, A., B. Boots, and K. Sugihara (1992). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, John Wiley and Sons.

O'Rourke, J. (1998). *Computational Geometry in C*, Cambridge University Press, 2nd ed.

Otsu, N. (1979). "A threshold selection method from gray level histograms." *IEEE Trans. Systems, Man and Cybernetics*, Vol. 9, pp. 62–66.

Parker, J. R. (1997). *Algorithms for Image Processing and Computer Vision*, Wiley Computer Publishing.

Pless, R. (2003). "Voronoi diagrams and delauney triangulations, Lecture 17, Computational Geometry." [On-line] 28 February 2006, http://www.cs.wustl.edu/∼pless/506/l17.html.

Quek, F. K. H., and M. C. Petro (1993). "Human-machine perceptual cooperation." in *CHI '93: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press, New York, NY, USA, pp. 123–130.

Reinders, F., M. Jacobson, F. Post, and E. Association (2000). "Skeleton graph generation for feature shape description." in *Data Visualization 2000*, Springer-Verlag Wien, pp. 73–82.

Rosenfeld, A., and A. C. Kak (1982). *Digital Picture Processing*, Academic Press, 2nd ed.

Russ, J. C. (2002). *The Image Processing Handbook*, CRC Press, 4th ed.

Serra, J. (1982). *Image Analysis and Mathematical morphology*, Academic Press, New York.

Sharma, O., D. Mioc, and A. Habib (2005). "Road extraction from satellite imagery using fractals and morphological image processing." in *Proceedings of the 13th International Conference on Geoinformatics*, Toronto, Canada.

Sibson, R. (1977). "Locally equiangular triangulations." *The Computer Journal*, Vol. 21, No. 3, pp. 243–245.

Siddiqi, K., S. Bouix, A. Tannenbaum, and S. W. Zucker (1999). "The hamilton-jacobi skeleton." in *International Conference on Computer Vision*, Corfu, Greece, pp. 828–834.

Sonka, M., V. Hlavac, and R. Boyle (1999). *Image Processing, Analysis, and Machine Vision*, PWS publishing.

Su, P., and R. L. S. Drysdale (1995). "A comparison of sequential delaunay triangulation algorithms." in *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*, ACM Press, New York, NY, USA, pp. 61–70.

Tanemura, M., T. Ogawa, and N. Ogita (1983). "A new algorithm for three dimensional voronoi tessellation." *Journal of Computational Physics*, Vol. 51, pp. 191–207.

Telea, A., and J. J. van Wijk (2002). "An augmented fast marching method for computing skeletons and centerlines." in *VISSYM '02: Proceedings of the symposium on Data Visualisation 2002*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp. 251–260.

Vincent, L., and P. Soille (1991). "Watersheds in digital spaces: An efficient algorithm based on immersion simulations." *IEEE Transactions on Pattern Analysis Machine Intelligence*, Vol. 13, No. 6, pp. 583–598.

Vlassis, N., Y. Motomura, and B. Kröse (2000). "Supervised linear feature extraction for mobile robot localization." in *IEEE International Conference on Robotics and Automation*, pp. 2979–2984.

Zhou, Y., and A. W. Toga (1999). "Efficient skeletonization of volumetric objects." *IEEE Transactions on Visualization and Computer Graphics*, Vol. 5, No. 3, pp. 196–209.

Appendix A

Matlab code

Following Matlab script is used to generate EPS images from VGUI exported ASCII files. This script needs to be run from within Matlab application. It asks the user to locate the ASCII coordinate file. It generates an EPS image with the same name as the input file but extension changed to `.eps` in the same folder.

```
1  %Author: Ojaswa Sharma
2  %E-mail: ojaswa_sharma@rediffmail.com

4  clc;clear;
5  [FileName,PathName] = uigetfile('*.*','Select an ASCII VGUI
       vector file');
6  file=strcat(PathName, FileName);
7  fid=fopen(file,'r');
8  description=fgetl(fid);
9  fgetl(fid);
10 img_dim_str = fgetl(fid);
11 img_dim=str2num(img_dim_str);
12 fgetl(fid);fgetl(fid);fgetl(fid);

14 %read the data now
15 data = fscanf(fid, '%f %f %f %f', [4 inf]);
16 Xdata=[data(1,:);data(3,:)];
17 Ydata=[data(2,:);data(4,:)];

19 %In the crust obtained from VGUI, image border is present
20 %Comment the following lines while processing skeletons

22 %****Remove image border****
```

```
idx=[find(Xdata(1,:)==0) find(Xdata(1,:)==img_dim(1)-1)];
idy=[find(Ydata(1,:)==0) find(Ydata(1,:)==img_dim(2)-1)];
id=[idx idy];
id=sort(id);
Xdata(:,id)=[];
Ydata(:,id)=[];
%****Done removing image border****

%plot the data
figure; hold on; axis equal; axis off
set(gca,'YDir','reverse');
set(gca,'XLim',[0 img_dim(1)],'YLim',[0 img_dim(2)])
%Plot extents for eps figures to show correct image extent
plot(0,0,'w');plot(img_dim(1),img_dim(2),'w');

line(Xdata,Ydata,'Color','k');%,'LineWidth',4
[pathstr, name, ext, versn] = fileparts(file)
if(isunix)
    sep_str = '/';
else
    sep_str = '\';
end

%Save as eps and close the figure
saveas(gca,strcat(pathstr,sep_str, name,'.eps'),'epsc2');
close
```

Vita

Candidate's full name: Ojaswa Sharma

University attended (with dates and degrees obtained):

Indian Institute of Technology (IIT) Roorkee, India — September 2003
Bachelor of Technology, Civil Engineering

Publications:

Sharma, O., Mioc, D., Anton, F., Dharmraj, G., 2005. Traveling salesperson approximation algorithm for real road networks. Advances in Spatio-Temporal Analysis, ISPRS book series, Taylor & Francis. (*In process*)

Sharma, O., Mioc, D., Anton, F., Dharmaraj, G., 2005. Traveling salesperson approximation algorithm for real road networks, International Journal of GIS. (*Submitted*)

Sharma, O., Mioc, D., Habib, A., Anton, F., 2005. Road extraction from satellite imagery using fractals and morphological image processing, submitted to Geomatica. (*Submitted*)

Conference Presentations:

Sharma, O., Mioc, D., Anton, F., Dharmaraj, G., 2005. Route Optimization in Transportation Networks. GIS Planet 2005, Estoril, Portugal.

Sharma, O., Mioc, D., Anton, F., Dharmaraj, 2005. Modified Christofides algorithm for Transportation Networks, XI Encuentros de Geometra Computacional, Santander, Spain.

Sharma, O., Mioc, D., Habib, A., 2005. Road extraction from satellite imagery using fractals and morphological image processing. 13th International conference on

Geoinformatics, Toronto, Canada. (*Best student paper award*)

Sharma, O., Mioc, D., Anton, F., Dharmraj, G., 2005. Traveling salesperson approximation algorithm for real road networks. ISPRS WG II/1,2,7, VII/6 International Symposium on Spatial-temporal Modeling, Spatial Reasoning, Spatial Analysis, Data Mining & Data Fusion (STM'05), Beijing, China.

Sharma, O., and Shah, N., 2002. Linear feature extraction from satellite imagery using fractals. Amateur paper presentation competition, Indian Institute of Technology, Roorkee, India.

Sharma, O., and Shah, N., 2002. Steganography - New frontiers in data hiding. Paper presentation competition, Birla Institute of Technology and Science, Pilani, India.