INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Université d'Ottawa · University of Ottawa

An Object Oriented Environment for Linear Feature Extraction from Aerial Images

by

Mihai A. Sasarman

A thesis submitted to the
School of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Applied Sciences

Ottawa-Carleton Institute of Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
University of Ottawa

To my Mother

ABSTRACT

We propose a new method for the detection of road networks in SPOT[1] images, with possible applications in automated cartography. This method is based on the integration of image segmentation techniques, in particular the watershed and Duda transformations, in a parallel Single Instruction Stream Multiple Data Stream (SIMD) environment. The Duda Road Operator provides the marking function for the watershed transformation of the image. Examples involving SPOT images are used and the resulting road networks are shown.

[1]Systeme Probatoire d'Observation de la Terre

ACKNOWLEDGMENT

I wish to express my sincere gratitude to my supervisor, Dr. Dan Ionescu, for his constant guidance, encouragement, and support throughout my research. I would like to thank my thesis co-supervisor, Dr. David Goodenough, for providing me with expertise and direction. In addition, thanks go to Ko Fung at the Canadian Centre for Remote Sensing for providing the aerial images used in my research.

I also wish to thank my colleagues and friends at Bell-Northern Research, and the faculty and staff of the University of Ottawa Department of Electrical Engineering, for their help and encouragement. I would also like to acknowledge the financial support from the University of Ottawa, Bell-Northern Research, and NORTEL.

I would further wish to thank Mr. Greg Clites and the staff at Applied Intelligent Systems Inc. in Ann Arbor, Michigan, for their support and encouragement throughout my research, and for providing the AIS-3500 parallel image computer on which I developed the GeoFind image processing system.

On a personal note, I would like to express my deepest gratitude to my fiancée, Martine, for her unwavering love and support, and to my family, for their support and encouragement. Most of all, I wish to express my thanks to my mother, to whom this thesis is dedicated, for her unconditional love and steadfast belief in my abilities. Through all their love, wisdom, personal example, and constant support, they made all this possible.

Contents

List of Figures

Notation

Various symbols and abreviations used throughout this thesis are introduced below. All notations are fully defined where first encountered in the text.

Symbols

C(M) Catchment Basin associated with a minimum M.

$d_A(x, y)$ Geodesic Distance between two pixels x and y in A.

$\mathcal{F}$(u) DRO function which measures the contrast between the road candidate and the adjacent areas (b_1, b_2, b_3) and (c_1, c_2, c_3).

$\mathcal{G}$(u) DRO function which measures the uniformity of the intensity along a road candidate (a_1, a_2, a_3).

$iz_A(B_i)$ Geodesic Influence Zone of a connected component B_i of B in A.

$SCORE$ a scoring function given in equation 3.2. for the detection of linear features.

$SKIZ_A(B)$ Skeleton by Influence Zones of B inside A.

Acronyms

apar Anti-Parallel.

CB Catchment Basin.

CCRS Canadian Centre for Remote Sensing.

DEM Digital Elevation Model.

DRO Duda Road Operator.

GIS Geographic Information System.

HRV SPOT High Resolution Visible sensor.

LANDSAT Land Satellite.

LDIAS LANDSAT Digital Image Analysis System.

MSS Multispectral Scanner.

NASA National Aeronautics Space Administration.

PE Processing Element.

pixel Picture Element.

SIMD Single Instruction Stream Multiple Data Stream.

SKIZ Skeleton by Influence Zones.

SLAP Scan Line Array Processor.

SPOT Système Probatoire d'Observation de la Terre.

TM Thematic Mapper.

Chapter 1

Introduction

In this chapter, the purpose and focus of the research is introduced. The problem of linear feature extraction is defined and the organization of the thesis is presented.

1.1 Overview

This thesis presents a new parallel implementation of an object oriented image processing system responsible for linear feature extraction from aerial images. This is accomplished through the help of a new end-to-end detection process which includes pre-processing, filtering, segmentation, and post-processing. This new system called GeoFind has been successfully applied to SPOT[1] imagery available from the

[1]Système Probatoire d'Observation de la Terre.

Canadian Centre for Remote Sensing (CCRS) located in Ottawa, Canada, for the purpose of road network detection.

1.2 Background

The subject of feature extraction from aerial images is of great interest to the research community. Analysis of aerial images is, in general, a complex task. A prime candidate for this perceived complexity is the presence of textures which cause difficulties for low-level processes such as filtering and segmentation. In addition, features of interest may be small compared to the size of the complete image, which may render them difficult to distinguish from background noise.

The following factors determine the amount of information that can be extracted from an image:

- the degree of detail available in an image, which depends on the following:
  - the scale or resolution of the image.
  - the contrast amongst distinct features in the image.
  - the spectral range of the image.
- the data extraction method.
- the skill of the interpreter.
- the characteristics of the features of interest.

Feature characteristics can be either *spatial*, such as pattern, texture, size, and shape, or *spectral*, such as intensity (i.e. single image) or color (i.e. multiple image).

The degree of difficulty encountered when extracting features from aerial images depends on the following factors:

- ◇ the information is presented in a highly compact fashion.[2]
- ◇ a certain degree of noise is present which is sometimes indistinguishable from the required information.
- ◇ objects in the scene are not clearly separated (i.e. no clear boundaries are immediately apparent).
- ◇ the extraction of the required features from among the wealth of information presents an overwhelming task, due to various factors which include: the large amount of computational data available in the scene, the presence of objects which obstruct the features of interest, absorption and reflectivity of light by unlike surfaces, climatic conditions, the resolution of the satellite image system, etc.

Photo interpretation is the process of extracting enough information from an image to create a meaningful map representation. This process makes use of structural

[2]A satellite with suitable spatial resolution is the French satellite, Système Probatoire d'Observation de la Terre (SPOT). The SPOT High Resolution Visible (HRV) sensor in panchromatic, known as PLA, has a spatial resolution of 10 meters. The Canadian Centre for Remote Sensing (CCRS) produces a geocoded digital product that is re-sampled at 6.25 meters. The single band imagery is collected over the 0.51 to 0.73 micrometer spectral region.

and spectral information. The *photo interpreter* interprets the input image information by using domain-specific knowledge pertaining to a particular geographic feature such as context, texture, spectral intensity, and structure.

The process of photo interpretation is iterative, involving various viewpoints such as global, local, domain-independent, domain-dependent, contextual, and textural. This process is:

time consuming It takes hours, sometimes days to interpret a large aerial image.

biased The same image interpreted twice by the same person or different persons will not necessarily yield the same results.

Thus, the justification for an automated linear feature extraction mechanism which supports the user in identifying and mapping the features of interest.

To reduce the amount of resources required for the large computational task inherent in the interpretation of large aerial images, a new *parallel* implementation is proposed using a massively parallel image computer on a single instruction multiple data (SIMD) architecture.

1.3 Problem Definition

This thesis focuses on the design and implementation of a new *automated* linear feature extraction system with practical application in the extraction of road

networks from aerial images in a Geographic Information System (GIS) environment. The system presented in this thesis has been designed using object oriented techniques and has been implemented on a massively *parallel* architecture.

Object oriented environments greatly enhance programming productivity. Benefits of object oriented techniques include:

software reusability With object oriented techniques one can enhance and modify existing library objects.

modularity Objects are organized as *black boxes* with well defined message interfaces. Changes in the design of an object are made without modifying code that uses the object, provided that the message interface is preserved.

organization Hundreds of operations available in object libraries are presented as a few dozen objects plus the operations for those objects, enhanced by a well structured class hierarchy.

The object oriented environment used in the detection of linear features is described in Chapter 4.

An *automated* linear feature extraction system must meet the following criteria:

robustness The system must exhibit a high degree of success with aerial images of differing quality.

reliability The system must provide results with a high degree of dependability.

repeatability The system must be able to extract the linear features in the same scene consistently.

accuracy The results of the extraction mechanism must offer a high degree of accuracy.

ease of use The learning curve for the system must be short.

flexibility The user must have maximum flexibility and control over the outcome of the results.

user friendly The system user interface must be designed in such a way as to allow the user to use the system efficiently.

An *automated* linear feature extraction technique is important as a digital aid for remote sensing in the following aspects:

- it may help speed up information collection for transportation, planning, and management.
- system detectable control points can be used for geometric correction.
- for high resolution imagery, patterns of linear features may provide some very useful contextual information to help identify residential areas, commercial areas, and industrial complexes.

The new automated linear feature extraction system developed for this research

receives SUN raster[3] format SPOT aerial images and outputs the road map to be used in the update of maps generated through Geographic Information Systems (GIS).

The system employs the following image processing modules: pre-processing, filtering, segmentation, and post-processing.

The implementation of the modules has been developed using new parallel algorithms. This approach provides a solution to the problem of applying existing recursive techniques to very large images containing thousands by thousands of pixels[4] with every pixel containing potentially more than one image channel. These well known techniques make significant demands for computation time and internal main memory. The parallel implementation introduced in this thesis addresses these concerns by using parallel programming techniques to process the image through a number of PEs[5] . For the purposes of this research, an AIS-3500 massively parallel image computer with 128 PEs has been used. Each PE is a general purpose bit serial processor which can perform boolean, arithmetic, and neighborhood operations.

1.4 Organization of the Thesis

The remaining chapters are organized in the following manner.

[3]A standard SUN image file format.

[4]Picture element.

[5]Processing elements.

Chapter 2 gives a review of the body of literature on the topic of linear feature extraction from aerial images. A classification of road extraction methods is also included.

Chapter 3 presents the new major algorithms developed within the scope of this research. These are the parallel DRO[6] filter and the parallel watershed transformation.

In Chapter 4, the parallel DRO filter output, used as a marking function, is combined with the parallel watershed algorithm to obtain the desired segmentation of the linear features of interest. The new image processing system called *GeoFind*, used in linear feature extraction is introduced. In addition, Appendix A gives the parallel implementation in the Intelligent C object oriented language of all the algorithms used in the image processing system.

In Chapter 5, the results of the application of the linear feature extraction algorithms to the detection of road networks in an urban area are presented. A discussion of the results together with an analysis of the robustness of the system follows.

Finally, in Chapter 6, a summary of the entire volume of research is presented. A discussion follows with regards to additional work which can further the research in automatic linear feature extraction.

---

[6]Duda Road Operator.

Chapter 2

Aerial Image Interpretation

This chapter presents a review of the literature in the area of image interpretation with particular attention paid to linear feature extraction approaches.

2.1 Introduction

Man's remote sensing of the earth, using instruments other than the naked eye, began in 1859 with Gaspard Tournachon's photograph from a balloon of a village near Paris, France.

Prior to 1960, remote sensing, although not named as such, was carried exclusively with photographic cameras. Aerial photography was given a major boost during World War I by the need for military reconnaissance, but it was World War II that led to widespread application of photo reconnaissance and instigated the train-

ing of large numbers of photo interpreters. Other technologies such as radar systems and thermal infrared devices owed their development to military needs. With the advent of powerful computers, aerial image interpretation received a great deal of attention from the research community. However, the major development that affected civilian remote sensing the most, prior to 1960, was the development of aerial color and color infrared photography. The formation of the National Aeronautics Space Administration (NASA) in 1958, and the development of the early planetary exploration programs, was primarily responsible for the development of the modern optical remote sensing systems as we know them today.

A major milestone was the development of the LANDSAT Multi-spectral Scanner (MSS) since it provided the first multi-spectral synoptic images in digital form. The first LANDSAT-1 was launched in 1972. The Thematic Mapper (TM), a high spatial resolution imaging, multi-spectral radiometer was first launched on LANDSAT-4 in 1982, and subsequently on LANDSAT-5 in 1984. The four spectral channels of the MSS were increased to seven in the TM with the inclusion of two short-wavelength and a thermal infrared channel. The NASA recent projects in the United States provide for an increase of the channel number to 200.

Other examples of remote sensing programs are the Systeme Probatoire d'Observation de la Terre (SPOT) program which consists of a series of earth-observing missions that are being undertaken by the government of France.

Today we are using images taken from orbit throughout the electromagnetic spectrum extending from the ultraviolet to microwaves. Goetz [23] restricted the

uses of remote sensing to coverage of the optical region generally considered to extend from 0.4 to 1000 micrometers, but restricted further by atmospheric transmission to windows within the 0.4 to 15 micrometer region.

In all applications of remote sensing data, the useful parameters are derived from radiometry, or in other words, the determination of the reflected or emitted radiance from the surface as a function of wavelength and position. The electromagnetic energy available for passive remote sensing, is derived from the sun, either as reflected sunlight or re-emitted thermal radiation. Examination of the earth's surface via the emission and reflection of the electromagnetic radiation is greatly complicated by the intervening atmosphere. Radiant energy and the atmosphere interact through scattering and absorption, and the amount of energy that the atmosphere either removes from or adds to that emitted or reflected from the surface depends on:

- path length — a function of the relative geometry of the source, surface and sensor.
- the reflectance of the surface surrounding the scene being viewed.

In summary, the major effects of the atmosphere are:

- limitations on wavelengths at which observations can be made.
- elimination of large areas by cloud cover.

- spatially and temporally variable contamination of the available data due to aerosol scattering.

Linear planimetric features can be defined by their major centre axis and treated as linears. Humans are very good at identifying features of interest, but digitizing is a very tedious and time-consuming operation. Research in the area of computer vision has shown how various low level operators can assist the photo interpreter in recognizing features of interest.

The subject of planimetric mapping is a challenging one in several respects. The amount of information present in aerial images, be it from LANDSAT, SPOT, the space shuttle, or other means, is huge, and thus the process of extracting meaningful information from these images, very tedious indeed. A prime cause is the presence of texture which causes difficulties for the low level processes such as edge detection and segmentation. Another source of difficulty is that the objects of interest may be small compared to the size of the complete image. Specific structures of interest may have special properties, known *a priori*, which allow for their easy extraction.

Given the problem of producing an overlay showing clearly visible roads in an aerial image, a person would normally be expected to accomplish this task with little difficulty, even though he or she may be completely unfamiliar with the terrain depicted in the image.

Generally speaking, the process of image understanding by computer can be divided into two major levels of information processing:

low level processing Transform an image to enhance some detail or desirable property, or classify an image into predetermined categories.

high level processing Perform semantic analysis of image patterns based on the features abstracted from low level processes.

The major difficulties with computer image understanding are:

- an image under-constrains a scene. It does not provide enough information by itself to recover the scene.
- it is difficult to extract some image features such as shape characteristics and spatial relationships from a digital image and represent them in abstract form convenient for computer processing.
- vision is easy for humans, but the knowledge in human vision is not well defined. Essentially, remote sensing image understanding is a process of abstraction of information from remotely sensed images and of application of domain knowledge to identify ground objects.

Completeness and explicitness of *factual* information and domain knowledge are critical for any knowledge-based problem solving. To obtain correct results, it is important to acquire all the necessary information and domain knowledge and represent them as suitable abstract forms for computer processing. As noted, an image usually cannot directly provide sufficient information for computer image

understanding — some image features are difficult to extract and describe, and image understanding knowledge is difficult to formulate. In addition, remotely sensed data is sometimes distorted and degraded, and small features in satellite images with relatively low spatial resolution are difficult to detect.

Research whose object has been to cope with these limitations and offer solutions to the problem of linear feature extraction from aerial imagery is reviewed below.

2.2 Classification of Road Extraction Methods

A variety of methods has emerged for the extraction of linear features from digital satellite data, with application to the detection of road networks [50] [25] [13] [30] [8] [41] [44]. These include edge and line filtering techniques [36] [26], contextual filtering [6] [50] [27] [28] [29], rule- and knowledge-based algorithms [20] [57] [24] [48], and mathematical morphology [18] [15] [39].

The importance of effective early processing of visual input for a complete understanding system is generally accepted. The early processing may consist of a description of image discontinuities, usually in the form of edges and lines, or segmentation into uniform regions. Nevatia and Babu [36] described a technique for extracting linear features in an image by a process of edge detection and line linking, and also of deriving higher level description from the extracted lines. The process of line finding consists of determining edge magnitude and direction by convolution

of an image with a number of edge masks, of thinning and thresholding these edge magnitudes, of linking the edge elements based on proximity and orientation, and finally of approximating the linked elements by piecewise linear segments. Roads are characterized by being bordered by nearly parallel line segments of opposing contrast known as *anti parallel* segments or *apars*. Adequate linking can be obtained simply by considering connections of an edge point with its 8-neighbors.

Nevatia and Price [42] used edge detection techniques to detect linear objects in the model representation of a scene. In this representation, a general specification of the objects was first created as a sketch. From the information on the sketch, an internal description was generated. This internal description is a graph structure with objects and feature values at the nodes and relations between objects as the arcs. The nodes in the graph of the model represent several different types of objects. They can be simple, complex, or generic objects.

The basic edge detection was performed by convolving a given image with a number of masks corresponding to ideal step edges in several directions. Binary edges were obtained by thinning and thresholding the edge magnitudes. These edges were then linked to their neighbors based on proximity and similarity of orientation. Finally the linked boundary segments were approximated by piecewise linear segments. Some specific structures — i.e. roads, that are known *a priori* to be bounded by locally linear and parallel boundary segments of opposite contrast, were located.

Guindon [26] applied a modification of this technique to multi-temporal LAND-

SAT thematic mapper (TM) data to extract road networks. Multi-temporal editing of the extracted road networks solved the problem of transient features in the imagery (i.e. poor spectral contrast in a scene).

Venkateswar and Chellappa [51] described a labeling technique which produces line descriptions from aerial images. The line extraction algorithm has five stages. These include edge detection, edge pixel linking, line extensions based on continuity, line merging based on collinearity, and line thresholding. The input to the system was in the form of an edge image produced by the Canny edge detector [12]. The edge direction of each edge pixel was quantized into one of eight directions. At the end of this step, each edge pixel had a line label associated with it. Edge pixels which do not belong to a line could be used to extend lines found in their neighborhood. A further refinement merged collinear lines while suppressing, so called, *noisy* lines through heuristic criteria.

Using spatial relationships (i.e. distance, direction, connectivity and containment) within a LANDSAT multi-spectral scanner (MSS) image. Gurney [27] applied a non-linear line detecting algorithm with optimal threshold selection based on contextual information to extract rivers and roads. A contextual algorithm was applied to fill the gaps and suppress noise subsequent to line detection. Bajcsy and Tavakoli [6] tracked segments to an end and then performed a search over a given area for a new segment. Vanderbrug and Rosenfeld [50] used contextual information related to the size, orientation and position of line segments to locate and join associated linear features.

The recovery of object shapes from the result of low level image analysis is a key problem for advanced vision systems. This is particularly true for aerial images, where scene objects correspond directly to uniform textured regions with known generic shapes. Lipary *et al.* [35] used robust curve-fitting to extract axially generated clusters — i.e. extended sets of related points that follow an axis curve. The emphasis was on elongated regions as observed from curvilinear alignments and proximities of representative point features. Curve model and scale dependency allowed the technique to decompose an arbitrary region segmentation into partitions with attributes corresponding to a specific object type. This analysis was performed first by computing those sets meeting general model constraints, detecting the discontinuities of analytical shapes within these sets, and then curve fitting to give piecewise smooth delineations.

In general, curve-fitting approaches to pattern recognition are based on constraints of spatial relationships defined over partitions of elements. Specific curve models are fit for the ability to achieve piecewise smooth descriptions of the low-level segmentation of specific scene objects.

Nicchiotti and Ottaviani [37] developed an automatic curve detection approach which avoided the traditional edge detection and linkage, by allowing to investigate the curve as a whole and not as a pixel sequence. Thus, it became possible to take into account the continuity information inherent in linear features that edge detectors do not exploit. The snakes technique, introduced by Kass at al. [33] for stereo matching and motion tracking, takes into account properties like continuity

and smoothness, and has, to some extent, the capability to fill in sections that have been occluded. The force that attracts the snake[1] to the feature is the minimization of the curve energy, searching a low cost curve between two end points.

Hu *et al.* [31] presented a semi-automatic method for tracing road networks in aerial images, through the use of an interactive system which allowed a road to be selected by picking its end points. The system then traced the road between the end points by searching for a shortest path defined by metrics based on well-known properties of road features.

Wang *et al.* [56] introduced an algorithm called Gradient Direction Profile Analysis (GDPA) used to extract road networks digitally from SPOT High Resolution Visible (HRV) panchromatic data. The technique is most effective in areas where road development is relatively recent. This is due to spectral consistency of new road networks. Additional noise removal and thinning algorithms were used to obtain maps of road networks.

Ogier *et al.* [40] introduced a semi-automatic road network extraction process which included a line following algorithm connected with a validation test based on a Gibbs distribution.

Schanzer *et al.* [44] discussed research efforts on the delineation of urban street networks from satellite data using the Duda Road Operator (DRO). A road is primarily a ridge that is flat; this is basically what the DRO measures. By relying on

[1]The continuous curve that attempts to position itself starting from a condition on the linear feature of the image — typically set to a straight line segment.

the human visual system to indicate whether a particular filter enhances or reduces the information content, a filter for each enhancement component was developed.

Hough has described an ingenious method for replacing the original problem of finding collinear points by a mathematical equivalent problem of finding concurrent lines. This method involves transforming each of the figure points into a straight line in a parameter space — i.e. the two-dimensional slope-intercept plane. Unfortunately, both the slope and the intercept are unbounded, which complicates the application of the technique. Duda and Hart [17] suggested an alternative parameterization that eliminated this problem. A normal parameterization specifies a straight line by the angle ϕ or its normal and its algebraic distance ρ from the origin. If one restricts ϕ to the interval $(0, \pi)$, then the normal parameters for a line are unique. With this restriction, every line in the x - y plane corresponds to a unique point in the $\phi - \rho$ plane. Suppose now that there exists a set of n points which have to be fit to a set of straight lines. If the points are transformed into sinusoidal curves, the curves corresponding to collinear figure points have a common point of intersection. Thus the problem of detecting collinear points can be converted to the problem of finding concurrent curves. The general transform approach can be extended to curves other than straight lines.

Fisher and Highnam [19] presented an algorithm for the Hough transform on a scan line array processor (SLAP) — i.e. an SIMD linear array of simple word-parallel processing elements (PE's). The Hough transform can be regarded as a set of projections of the boundary points along all lines of a chosen resolution. Thus by

traversing the route of a single line across the image one can calculate its support by incrementing a counter every time a boundary point is visited. When the line route being traversed terminates at the image border, the support for the line is known.

Burns *et al.* [11] presented the development of mechanisms for extracting straight lines from complex images, including lines of arbitrarily low contrast, and the construction of an intermediate representation of edge/line information through which high level interpretation mechanisms had efficient access to relevant lines. Curved lines can be approximated reasonably well as aggregates of piecewise linear segments.

The edge maps resulting from the application of a local edge detector are usually very dense and do not distinguish between edges resulting from object boundaries, shadows, and changes in surface reflectance and orientation. Fine detail (high frequency, subdivisions, etc.) image events respond optimally to different size operators. As a result, the appropriate size operator must be determined in each different area of the image.

In many cases, the local operators misplace or entirely miss edges, a single real edge may result in several strong operator responses at different, often parallel locations, and underlying data may not conform to expectations built into the grouping process. Low-contrast lines, because of their low signal-to-noise ratio, are often troublesome. The edge orientation carries important information about the set of pixels that participate in the intensity variation that underlines the straight line, particularly its spatial extent.

Wang and Newkirk [53] described research to develop a technique that extracted highway networks, located highway intersections, and identified different road patterns. Problem complexity in the higher level processes was reduced by obtaining as good as possible an output from the low level processes. The analysis and processing were performed iteratively. At each iteration, individual linear patterns were checked to see whether extension or connection with others was possible. If a linear pattern could not be extended or connected, the average intensity values of the pattern and its surrounding areas, and its connection and contextual information were analyzed to decide at which confidence level it could be labeled as a highway segment.

Zlotnick and Carnine [59] described a road finding system which used clues in the form of anti-parallel edge pairs or *apars* and, by using a special algorithm to analyze noisy road center sequences, spanned gaps in these clues to construct road hypotheses even when edge continuity was imperfect. Further improvement of results was achieved by linking road hypotheses that were consistent in a geometric sense.

Barzohar and Cooper [7] presented an automated approach to finding main roads in aerial images by building geometric-probabilistic models for road image generation. Given an image, roads were found by map estimation. This was handled by partitioning an image into windows, realizing the estimation in each window through the use of dynamic programming, and then, starting with the windows containing high confidence estimates, using dynamic programming again to obtain op-

timal global estimates of the roads present. Road curvature, width, image intensity, and edge strength could vary considerably.

Zelek [58] developed a set of tools that assisted the feature extraction process. The tools extracted primitives (domain independent features). Primitives were primarily described by their shape and, secondly, by their spectral characteristics. The extraction of geometric features was based upon the extraction of edges in the imagery. The edge detection scheme was based on the Marr and Hildreth approach. Edge information provided clues for the locations of boundary features. The only implemented perceptual grouping of edge boundary segments was that of narrow anti-parallel line segments or *apars*. The user could select which *apars* to keep as road candidates and a simple algorithm was run to connect these candidates together based on a road model and proximity criteria.

Wang and Howarth [55] developed a linear-feature network detection and analysis system which was used in the identification and mapping of lineaments from satellite images, used in the exploration for mineral deposits, since lineaments may be related to ore deposits. On satellite images, lineaments usually appear as straight lines or edges, but frequently may have gaps due to poor contrast or coverage by surface materials. In visual interpretation and mapping, geologists use their knowledge and experience to connect lines and edges which are collinear and broken into a series of segments. Some of the rules used by geologists in their image interpretation were applied in automated lineament extraction from digital imagery by using an Edge Following as Graph Searching (EFGS) algorithm to trace all edges on the

image.

Given the problem of producing an overlay showing the clearly visible roads in an aerial image, a person would normally be expected to accomplish this task with little difficulty, even though she may be completely unfamiliar with the terrain depicted in the image. Fishler *et al.* [20], specified the requirements and mechanism for a machine to be capable of near human performance in finding roads and other semantically meaningful linear structures.

At low resolution, roads are line-like structures of essentially constant width, which, in general, are locally constant in intensity and show significant contrast with adjacent terrain (generally they are uniformly lighter or darker). A specific interpretation of this low-resolution road model is embodied in the Duda Road Operator (DRO).

The basic approach in the paper was to strongly bias (or even constrain) the desired answer to fit the coherent pattern produced by a superposition of evidence provided by all the type I operators and to fill in the details locally, using the particular type II operator which seemed to be most certain that it found the desired feature. This was synthesized into an algorithm for precisely tracking the major linear structure visible in a delimited region of an image. The Low-Resolution Road Tracking algorithm (LRRT), takes as input an image array, a search region defined by a binary mask, and constraints on starting and ending coordinates of the track.

Abrams *et al.* [1] used an algorithm to detect structural features having a

maximum local brightness gradient. A criterion of linearity was calculated which compared the total length of the contour to the Euclidean distance between the endpoints. This parameter had a value of 1 for a straight line, increasing with the sinuosity and curvature of the line. By selecting an appropriate threshold value, sinuous contours were largely eliminated.

Smith and Austin [45] described work aimed to evaluate the ability of neural networks for feature recognition. Their use in segmentation of textural regions looks attractive because the networks principally operate by collecting statistical correlations in the input images.

Nicolin and Gabler [38] aimed to develop and implement a complete system, starting from the digitized picture and leading to a consistent, unambiguous, and reliable labeling of essential image segments. The approach combined methods from image processing and artificial intelligence in trying to overcome the failure of the segment-and-label paradigm. The system worked on panchromatic aerial photographs. To enhance image quality, a diversity of pre-processing methods were applied interactively to the image as needed. Graylevel analysis was executed in order to get clues as to where segmentation could be successfully applied in the image, by systematically computing contrast features of the image. Segmentation was regarded as merely a mechanism to divide the image in meaningful parts.

Interpretation of a set of regularly arranged and similar image segments usually leads to less ambiguous, more reliable, and more consistent results than the interpretation of isolated items. Structural analysis detects regular arrangements

between image segments, taking advantage of certain principles: similarity, smooth continuation, proximity, symmetry, and familiarity.

Since 1973, the Canadian Centre for Remote Sensing (CCRS) has been conducting research and development into information extraction methods and systems. In 1982, the development of the LANDSAT Digital Image Analysis System (LDIAS) was begun. Systems for remote sensing analysis are usually complex. To simplify the operation of LDIAS, several expert system advisors have been developed. Goodenough *et al.* [24] described the approach to integrating LDIAS with expert systems. Two advisors were analyzed: an Analyst Advisor to guide the user during the analysis of LANDSAT MSS and TM imagery and a Map Image Congruency Evaluation (MICE) advisor. Both of the advisors utilized a shell developed at CCRS called Remote Sensing Expert System Shell (RESHELL).

In the map making and updating process, photo interpreters typically analyze aerial photographs, decide on the classification of different objects in the photograph, and then transcribe the classification and location of these objects onto a map or directly into a Geographic Information System (GIS). The map is only an approximation of the real world, prone to human error. A further complication is that the world land mass is a constantly changing entity, while cartographic data are relatively static — i.e. periodically updated. Thus the justification for integrating remote sensing data with GIS databases. However, the automatic integration of remote sensing data with GIS is not yet possible as human interpretation and assistance are still required [5]. The integration problem occurs when some area in

a remote sensing image is to be placed in a GIS. Database corruption will occur if there is not a perfect spatial juxtaposition of the image and the map.

In the field of image analysis and more particularly of mathematical morphology, graphs of gray tone pictures are often regarded as topographic surfaces, the graylevel of a pixel standing for its elevation. The whole set of points of the surface whose steepest slope paths reach a given minimum constitutes the catchment basin associated with this minimum. The watersheds are the zones dividing adjacent catchment basins. The first watershed algorithms were developed to extract terrain divides from Digital Elevation Models (DEMs). Meanwhile, the watershed transformation was studied in the field of image processing. It constitutes one of the most powerful tools for contour detection and image segmentation provided by mathematical morphology. Soille and Vincent [47] introduced a flexible implementation of this transformation based on a progressive flooding of the picture. Pixels were first sorted in the increasing order of their grayscale values. Then successive graylevels were processed in order to simulate the flooding propagation. The minima of the function were sorted in the increasing order of their graylevel values. Then, the basin of the minimum at the lowest altitude was flooded progressively. When the water reached the altitude of the second minimum, both basins were flooded simultaneously until the elevation of the third minimum was reached, and so forth. In addition, dams were erected at the places where the waters coming from two different basins would merge. At the end of this flooding procedure, each minimum was completely surrounded by dams, which delimited its associated catchment basins. The set of dams thus obtained corresponded to the watersheds, and provided a

tessellation of the image in its different catchment basins.

When combined with other tools provided by mathematical morphology, the watershed transformation constitutes a powerful procedure of contour detection and image segmentation. The difficulty is due to low contrast and high noise level. Simple computation of watersheds of its morphological gradient leads to considerable over segmentation. The search contours are lost in numerous irrelevant ones. To get rid of over segmentation at least two alternatives are available: either remove the irrelevant arcs of the watersheds by region growing — i.e. post-processing, or modify the gradient function in such a way that its watersheds coincide with the contours of the desired objects — i.e. pre-processing, by using *a priori* knowledge in the form of a marking step. These markers are used to preserve the watershed lines located between the markers. The computation of the watersheds of this modified gradient then provides the desired segmentation. The catchment basin of the background marker stands for the background itself whereas the watersheds correspond to the contours of the desired object. Thus getting a good segmentation involves determining a marker of the background and a marker for each geometrical shape. An intermediate solution consists in marking manually these objects, but an automatic extraction of the markers is preferred.

2.3 Conclusion

This chapter presented a review of the field of linear feature extraction from aerial images with particular emphasis on some of the more well known methods used for linear feature extraction.

The algorithms developed in this research will introduce a new parallel implementation of linear feature extraction using a combination of pre-processing methods (i.e. a marking step), parallel watersheds to extract the features of interest, and post-processing methods (i.e. region growing) using mathematical morphology.

Chapter 3

Parallel Algorithms for Linear Feature Extraction

This chapter describes the grayscale morphological transformation applied to the original aerial image which has the effect of eliminating the irrelevant information and preserving the linear features. A parallel Duda Road Operator algorithm will be introduced. This algorithm will be used as a marking function in the overall design approach. Secondly, the parallel watershed algorithm will be derived. Their implementation [43] will be discussed in Chapter 4.

3.1 Introduction

The first step in the linear feature recognition process is the calculation of the marking function through the use of the Duda Road Operator (DRO). Section 3.3

will first present a brief description of the general DRO algorithm followed by a discussion of the new parallel algorithm developed as part of this research.

In the second step, the marking function output will be used as the seed for the watershed transformation which will identify the linear features of interest, in our case, the road network. Section 3.4 will introduce the watershed transformation together with the new parallel watershed algorithm.

3.2 Parallel Algorithms for SIMD Architectures

Let $\mathcal{A} = [a_{ik}]$ and $\mathcal{B} = [b_{kj}]$ be $n \times n$ matrices which represent aerial images. An operation $\bigcirc$ between $\mathcal{A}$ and $\mathcal{B}$ generates a result image $\mathcal{C} = \mathcal{A} \bigcirc \mathcal{B} = [c_{ij}]$ of dimension $n \times n$. The elements of the result matrix or image $\mathcal{C}$ are related to the elements of $\mathcal{A}$ and $\mathcal{B}$ by:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \circ b_{kj} \text{ for } 1 \leq i \leq n \text{ and } 1 \leq j \leq n \qquad (3.1)$$

In the SIMD parallel vision computer with n PEs, each row vector of the matrix or image is stored across the PEs. Column vectors are then stored within the same PE as shown in Figure 3.1. This memory allocation scheme allows parallel access of all the elements in each row vector of the matrices.

In the parallel algorithms introduced in this chapter, all mathematical morphology operations are performed in parallel on each row of the image.

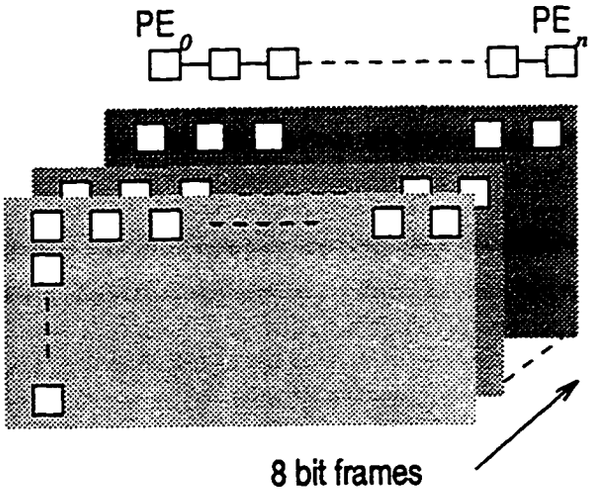

Figure 3.1: SIMD parallel vision computer with n processing elements (PEs).

3.3 The Duda Road Operator

The SPOT images are *marked* by a process which detects elongated features in the first step of the linear feature extraction. This marking step gives the starting point to the subsequent watershed transformation.

3.3.1 General Algorithm Description

The Duda Road Operator (DRO) [44] is an efficient filtering algorithm suited for the detection of elongated features. The DRO has been used by Fishler *et al.* [20] as a cost function for the detection of roads in low-resolution aerial imagery. The same operator has been used for linear feature detection in SPOT images.

The DRO combines a function $\mathcal{G}(u)$ which measures the uniformity of the in-

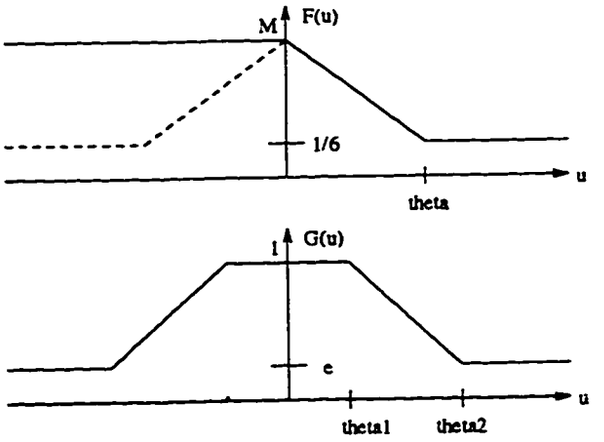

Figure 3.2: The functions $\mathcal{F}$(u) and $\mathcal{G}$(u) of the Duda Road Operator.

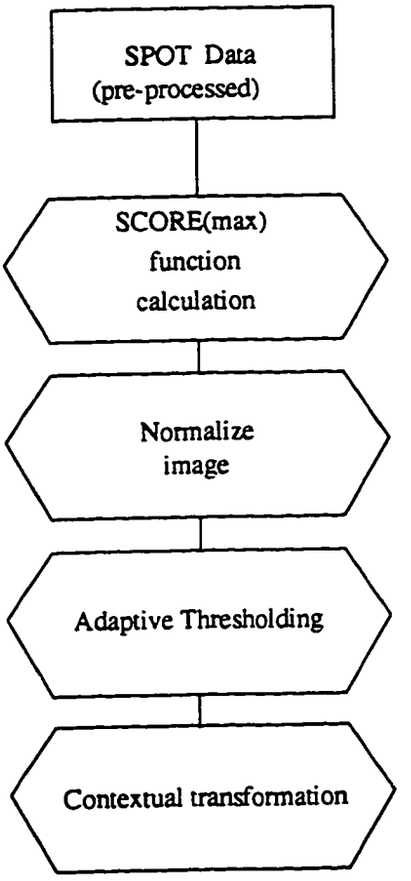

Figure 3.3: The Duda Road Operator algorithm.

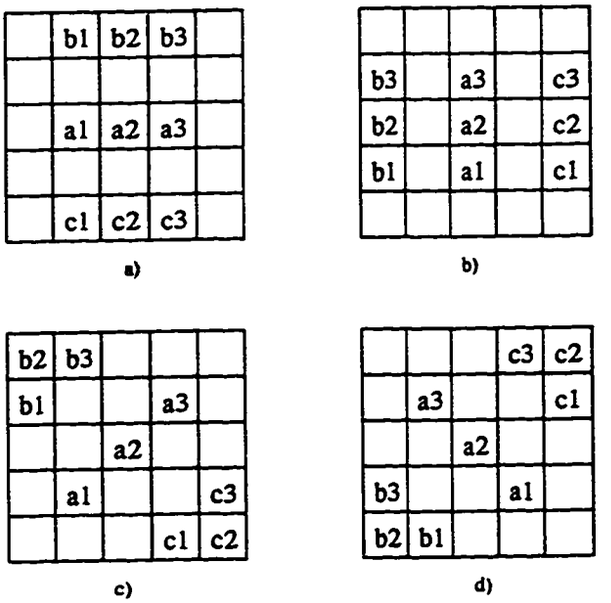

Figure 3.4: The four directional masks of the Duda Road Operator used to detect linear features: a) horizontal, b) vertical, c) right diagonal, and d) left diagonal.

tensity along a road candidate (a_1, a_2, a_3), and a function $\mathcal{F}(u)$ which measures the contrast between the road candidate and the adjacent areas (b_1, b_2, b_3) and (c_1, c_2, c_3), as shown in Figure 3.2. A separation of one pixel between the road candidate and the adjacent areas allows for small variations in road width.

The function $\mathcal{G}(u)$ ignores small (less than θ_1) and large (greater than θ_2) intensity variations along the road. Similarly, function $\mathcal{F}(u)$ outputs a constant value ϵ for contrasts greater than θ.

The DRO uses a scoring function $SCORE$, given in equation 3.2, for the detection of linear features. This function is calculated for each of the four masks shown in Figure 3.4, with the maximum value $SCORE_{max}$ replacing the intensity value of the pixel a_2. In this way, the DRO gives the same response for a thin line of

amplitude d and an edge of intensity $2d$, which is the desired response for a line following algorithm. Figure 3.3 shows the main steps of the Duda Road Operator algorithm.

At the same time, however, the DRO tends to identify a large number of elongated features, not all of which are features of interest. For this reason, the DRO is used as a marking function only, in order to provide the seeds for the subsequent recognition step. The linear feature extraction is performed by the watershed transformation, introduced in section 3.4.

Definition 3.1 *Let A be a set in Euclidean $\mathcal{N}$ space. For ordinary grayscale imagery, $\mathcal{N} = 3$. Let $A \subseteq E^N$ and SCORE be a function*

$$SCORE = \mid G(\mid a_1 - a_2 \mid) \times G(\mid a_2 - a_3 \mid) / \sum_{i=1}^{3} (F(a_i - b_i) + F(a_i - c_i)) \mid \quad (3.2)$$

where:

$$F(u) = \begin{cases} M & u < 0 \\ M - \frac{(M-\frac{1}{6})u}{\theta} & 0 \leq u \leq \theta \\ \frac{1}{6} & u > \theta \end{cases} \quad (3.3)$$

$$G(u) = \begin{cases} 1 & 0 \leq u < \theta_1 \\ 1 - (1-\epsilon)\frac{u-\theta_1}{\theta_2-\theta_1} & \theta_1 \leq u \leq \theta_2 \\ \epsilon & u > \theta_2 \end{cases} \quad (3.4)$$

Then the transformation of A, denoted by $T[A] : SCORE \longrightarrow E$, *is defined by*

$$T[A](x) = max\{SCORE \mid (x, SCORE) \in A\} \tag{3.5}$$

The parallel Duda Road Operator algorithm is introduced in section 3.3.2.

3.3.2 Parallel DRO Algorithm

The conventional implementation of the DRO consists of using the four direction masks (horizontal, vertical, right diagonal, left diagonal) shown in Figure 3.4 on the image at every point P such that the maximum intensity $SCORE_{max}$ replaces the value of a_2. In the parallel algorithm, the $SCORE_{max}$ function is calculated by first applying the $SCORE$ function in each direction on the entire image through add and shift operations, and then by replacing the grayscale value of each pixel with the maximum $SCORE_{max}$ value selected from the respective four direction $SCORE$ values for that pixel. The operations performed in the calculation of the $SCORE$ function are done in parallel across each image row. The parallel DRO algorithm is comprised of four parallel blocks which follow each other sequentially, as shown in Figure 3.3.

The parallel DRO algorithm is comprised of three steps which are detailed in subsequent sections:

1. calculate the value $SCORE_{max}$ in parallel across each image row for the entire

image and replace the grayscale value of each pixel with $SCORE_{max}$ respectively.

2. normalize the new image by setting the maximum pixel intensity to 255.

3. perform a parallel adaptive thresholding of the normalized image to separate the elongated features.

SCORE Function Calculation

To calculate the $SCORE$ function, morphological operations are employed in parallel across each row, on the entire image in each of the four directions.

For each direction, the value of $SCORE$ is calculated at each pixel location. The final grayscale value of each pixel is given by $SCORE_{max}$, given in equation 3.5.

The steps involved in calculating the value of $SCORE$ with the horizontal mask on the parallel SIMD architecture are outlined below. The same steps are used in all the four directions.

- Let $\mathcal{F}$(u) be given by equation 3.3. Let $\mathcal{G}$(u) be given by equation 3.4. Using the above equations create the look-up tables for the above two functions respectively.

- Let A_i represent the original grayscale image. Let A_j represent the original image shifted one pixel west. Let A_k represent the original image shifted one pixel

east. Let A_{G_a} represent the first result image. Let A_{G_b} represent the second result image. Let A_{G_u} represent the third result image. Then,

$$A_{G_a} = max\{A_i - A_j, A_j - A_i\} \tag{3.6}$$

$$A_{G_b} = max\{A_k - A_i, A_i - A_k\} \tag{3.7}$$

◇ Using the look-up table, calculate the output of the function $\mathcal{G}(u)$

$$A_{G_a} = G(max\{A_i - A_j, A_j - A_i\}) \tag{3.8}$$

$$A_{G_b} = G(max\{A_k - A_i, A_i - A_k\}) \tag{3.9}$$

◇ Multiply the two previous results, by using a SIMD parallel multiplication algorithm [32] (i.e. repeated additions with carry).

$$A_{G_u} = A_{G_a} * A_{G_b} \tag{3.10}$$

◇ Let A_i represent the original grayscale image. Let A_l represent the original image shifted two pixels north. Let A_m represent the original image shifted two pixels south. Let A_{F_u} represent the result image.

Similarly to the previous steps, by using shift operators, calculate the following:

$$A_{F_u} = \sum_{i=-1}^{+1} F(A_i - A_l) + \sum_{i=-1}^{+1} F(A_i - A_m) \tag{3.11}$$

◇ Divide equations 3.10 by 3.11, by using a parallel division algorithm [32] (i.e. repeated subtractions with carry).

$$SCORE_H = A_{G_u}/A_{F_u} \tag{3.12}$$

At the end of the above transformation, the result image contains $SCORE_H$ as the grayscale value of each pixel. Similarly, three other result images will be calculated in the other three directions, $SCORE_V$, $SCORE_{RH}$, and $SCORE_{LH}$ respectively. The final result image will set as the grayscale value of each pixel, the maximum of the above four direction $SCORE$ values (i.e. $SCORE_{max}$) replacing the grayscale value of each pixel as shown in equation 3.5.

Image Normalization

The second step, the normalization process, transforms the image by setting the maximum grayscale value of the image in the scene $SCORE_{MAX}$ to 2^{8-1} or 255. As a result, for each pixel P_i in the image, having pixel intensity $SCORE(max_i)$, calculate the new normalized intensity $P(i_N)$, given by

$$SCORE(max_N) = 255/SCORE_{MAX} \times SCORE(max_i) \tag{3.13}$$

Parallel Adaptive Thresholding

In the third and final step, the normalized image is segmented by using a parallel adaptive threshold filter. The filter performs a bimodal threshold operation by first histograming the image and secondly, selecting the threshold value which best separates the two modes.

A frequency histogram consists of an array of bins, one bin for each possible extracted pixel grayscale value, where bin i contains the number of pixels with grayscale value i. In a bimodal histogram, the threshold value used to segment the image is given by the bin number that best separates two modes, a higher (lighter) and a lower (darker) mode, respectively. The lower (darker) mode represents an estimated percentage of the scene which has pixel grayscale values lower than i. In the SIMD architecture, the above adaptive threshold operation is performed in parallel across each image row.

The result of the DRO which contains the thresholded binary image contains a set of linear features which represent the basis or seeds of the watershed transformation described in section 3.4.

3.3.3 Initial Conditions

From the previous section, we know that the function $\mathcal{G}(\mathrm{u})$ ignores small (less than θ_1) and large (greater than θ_2) intensity variations along the road. Similarly, function

$\mathcal{F}$(u) outputs a constant value ϵ for contrasts greater than θ. Therefore, we must ensure that the original image will contain linear feature data which is statistically distinct from background noise, and in particular, edges which are larger than θ.

3.3.4 Parallel DRO Stability

The parallel Duda Road Operator is invariant with respect to translation and scaling as shown in section 5.3.2.

Parallel DRO Filter Invariance to Rotation

To verify the invariance of the parallel DRO filter with respect to rotation two test images have been used in which lines at intervals of 5 degrees are drawn. The results of the parallel DRO filter on these test images are shown in section 5.3.1. These results confirm, that this filter is not invariant to rotation. However, the parallel algorithm shows good results for rotation angles of +/- 5 degrees.

A marked improvement in rotation invariance of the DRO filter can be observed if the number of directional masks is increased to eight from the present four. This would increase the visibility of diagonal linear features.

Stability of the DRO Filter

It is apparent that the parallel DRO filter does a good job most of the time but has some significant weaknesses. It is sensitive to

- ◇ road orientation in directions other than the four principal directions explicitly covered by the DRO masks,
- ◇ raster quantization effects,
- ◇ sharp changes in road direction, and
- ◇ certain problems with the adjacent terrain.

The parallel DRO filter uses a scoring function for the detection of straight lines as shown in equations 3.2 through 3.4. This function is applied in conjunction with four different masks identifying the four directions: vertical, horizontal, right diagonal, and left diagonal. The three parameters which affect the performance of the filter are:

- ◇ θ - linear feature contrast threshold (default value 15) — function $\mathcal{F}$(u) outputs a constant value ϵ for contrasts greater than θ.
- ◇ θ_1 - lower intensity threshold along the linear feature (default value 5) — the function $\mathcal{G}$(u) ignores small (less than θ_1) intensity variations along the road.

- θ_2 - higher intensity threshold along the linear feature (default value 15) — the function $\mathcal{G}(u)$ ignores large (greater than θ_2) intensity variations along the road.

Tests were performed with different values for the above parameters and the best results were obtained with the indicated default values. The most robust performance is exhibited with the above parameter set.

3.4 The Watershed Transformation

The following section introduces the watershed transformation and describes the parallel watershed algorithm.

3.4.1 Introduction

A *watershed* line in the classical term refers to a line which separates two areas. The two regions separated by this line are called *catchment basins*. In the field of image processing and more particularly in Mathematical Morphology (MM), grayscale pictures are often considered as topographic reliefs. In the topographical representation of a particular image I, the numerical value (i.e. grayscale value) of each pixel stands for the elevation at this point. Such a representation is extremely useful, since it first allows one to better appreciate the effect of a given transformation on the image under study. We know, for example, that an opening removes

some peaks and crest lines, whereas a closing tends to fill in basins and valleys. Furthermore, thanks to this representation, such notions as minima, catchment basins and watersheds can be well defined for grayscale images.

The first algorithms for computing watersheds are found in the field of topography. Topographic surfaces are numerically handled through *digital elevation models* (DEMs). These are arrays of numbers that represent the spatial distribution of terrain altitudes.

The above notions were also studied in the field of image processing. The introduction of the *watershed transformation* as a morphological tool is due to Digabel and Lantuéjoul [16]. Their data were piles of binary images representing successive thresholds of a bituminous surface relief whose drainability was to be studied. Later, a joint work of Lantuéjoul and Beucher led to the inversion of this original algorithm in order to extend it to the more general framework of grayscale images [9] [10]. Watersheds have been used in numerous grayscale segmentation problems.

Extracting watersheds from digital pictures is far from being an easy task. The purpose of this section is to introduce a parallel watershed algorithm based on using mathematical morphology to identify watershed points at successive thresholds. This algorithm is much faster than conventional sequential algorithms. Also, it behaves well in all the particular pixel configurations where many existing algorithms produce incorrect results.

For the sake of clarity, Section 3.4.2 deals with the definitions used in wa-

tersheds. The efficiency and advantages of the parallel watershed algorithm are discussed in Section 3.4.7.

3.4.2 The Watershed Transformation

Basic Definitions

Let I be a two-dimensional grayscale image whose definition domain is denoted as $D_I \subset Z^2$. I takes discrete (graylevel) values in a given range $[0,N]$, where N is an arbitrary positive integer:

$$I \begin{pmatrix} D_I \subset Z^2 & \rightarrow & \{0,1,\ldots,N\} \\ p & \mapsto & I(p). \end{pmatrix} \tag{3.14}$$

Let G denote the underlying digital square grid in four or eight connectivity. G is a subset of $Z^2 \times Z^2$.

Definition 3.2 *A path P of length l between two pixels p and q in image I is a $(l+1)$-tuple of pixels $(p_0, p_1, \ldots p_{l-1}, p_l)$ such that $p_0 = p, p_l = q$, and $\forall i \in [1,l], (p_{i-1}, p_i) \in G$.*

In the following, $l(P)$ denotes the length of a given path P and $N_G(p)$ denotes the set of the neighbors of a pixel p with respect to G : $N_G(p) = \{p' \in Z^2, (p,p') \in G\}$.

Definition 3.3 *A minimum M of I at altitude h is a connected plateau of pixels*

with the value h from which it is impossible to reach a point of lower altitude without having to climb:

$$\forall_p \in \forall_q \notin M, \text{ such that } I_q \leq I(p), \text{ and}$$

$$\forall P = (p_0, p_1, \ldots p_l) \text{ such that } p_0 = p \text{ and } p_l = q, \text{ then}$$

$$\exists i \in [1, l] \text{ such that } I(p_i) > I(p_0). \tag{3.15}$$

A minimum is thus a connected and iso-intensive area where the graylevel is strictly darker than on the neighboring pixels (the darker the pixel, the lower its value or elevation).

Next, the definition of *catchment basins* and *watersheds* is introduced.

Definition 3.4 *Let I be a grayscale image. The catchment basin $C(M)$ associated with a minimum M is the set of pixels p of D_I, such that a water drop falling at p flows down along the relief, following a certain descending path called the downstream of p, and eventually reaches M.*

The lines which separate different catchment basins build what is called the *watersheds* (or *dividing lines*) of I.

The catchment basins of an image I correspond to the *influence zones* of its minima. Thus, a close relationship exists between the binary *skeleton by influence*

zones and the watersheds. The notion of catchment basin, like that of minimum, is not a local one. In many cases, no local considerations can allow one to decide whether two pixels belong to the same catchment basin or not.

For real, continuous, derivable functions (the only possible plateaus of these functions are minima), the direction of the flow path at any point is defined almost everywhere by the opposite of the gradient's azimuth at this point. On the contrary, when dealing with digital functions, there exists no rule to set up the path a drop of water would follow. Therefore this intuitive approach to watersheds is not well suited to practical implementations.

Definitions more suitable to the formalization of catchment basins and watersheds in digital spaces and more oriented to algorithm design are presented in the next section.

Definition by "Immersion"

By analogy, each regional minimum of I has a pierced hole, this image being regarded as a topographic surface. The surface is then slowly immersed into a lake. Starting from the minima of lowest altitude, the water progressively fills up the different catchment basins of I. A *dam* is built at each pixel where the water coming from two different minima would merge. At the end of this immersion procedure, each minimum is completely surrounded by dams, which delimit its associated catchment basin. The whole set of dams which has been built thus provides a tessellation of I

in its different catchment basins. These dams correspond to the watersheds of I.

A more formal definition follows.

Definition 3.5 *I being the grayscale image under study, denote h_{min} the smallest value taken by I on its domain D_I. Similarly, denote h_{max} the largest value taken by I on D_I. In the following, $T_h(I)$ stands for the threshold of I at level h:*

$$T_h(I) = \{p \in D_I, I(p) \leq h\}. \tag{3.16}$$

Definition 3.6 *Let $C(M)$ denote the catchment basin associated with a minimum M and $C_h(M)$ the subset of this catchment basin made of the points having an altitude smaller or equal to h. Then,*

$$C_h(M) = \{p \in C(M), I(p) \leq h\} = C(M) \cap T_h(I). \tag{3.17}$$

As concerns the minima of I, $\min_h(I)$ refers to the set of points belonging to the minima at altitude h.

The definitions of the *geodesic distance* and the *geodesic influence zones* will now be introduced. Let A be a set which is first supposed to be simply connected.

Definition 3.7 *The geodesic distance $d_A(x, y)$ between two pixels x and y in A is the infimum of the length of the paths which join x and y and are totally included in A:*

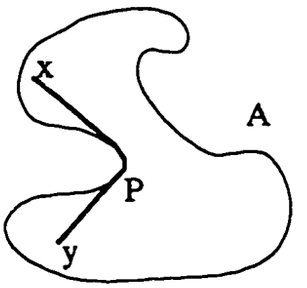

Figure 3.5: Geodesic distance between x and y in A.

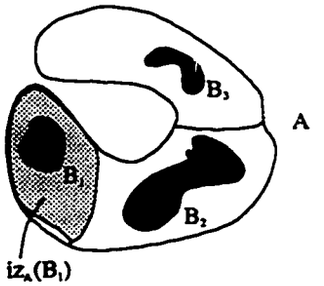

Figure 3.6: Geodesic influence zone of connected component B_1 inside A.

$$d_A(x, y) = inf\{l(P)\}. \tag{3.18}$$

where, P is the path between x and y which is totally included in A.

See Figure 3.5 for an illustration of the above definition.

Suppose now that A contains a set B made of several connected components $B_1, B_2, \ldots, B_k$.

Definition 3.8 *The geodesic influence zone $iz_A(B_i)$ of a connected component B_i of B in A is the locus of the points of A whose geodesic distance to B_i is smaller than their geodesic distance to any other component of B:*

$$iz_A(B_i) = \{p \in A, \forall j \in [1,k]/\{i\}, d_A(p,B_i) < d_A(p,B_j)\} \tag{3.19}$$

This concept is illustrated in Figure 3.6. Those points of A which do not belong to any geodesic influence zone constitute the *skeleton by influence zones (SKIZ)* of B inside A, denoted by $SKIZ_A(B)$:

$$SKIZ_A(B) = A/IZ_A(B) \tag{3.20}$$

where,

$$IZ_A(B) = \bigcup_{i\in[1:k]} iz_A(B_i)$$

According to this digital definition, the geodesic SKIZ of B in A does not necessarily separate the different geodesic influence zones. Indeed, due to parity problems, it is often made of *disconnected* lines. Moreover, the digital SKIZ may sometimes be a *thick* one, since the set of the pixels which are equally distant from two connected components may well be very thick. The parallel algorithm introduced in section 3.4.4 uses a labeling scheme of the catchment basins which avoids parity problems. The above definitions easily extend to the case where A is not simply connected, nor even connected at all. To simulate the immersion procedure described above, we start from the set $T_{h_{min}}(I)$, the points of which being those first reached by the water. These points constitute the starting point of the parallel algorithm. Thus:

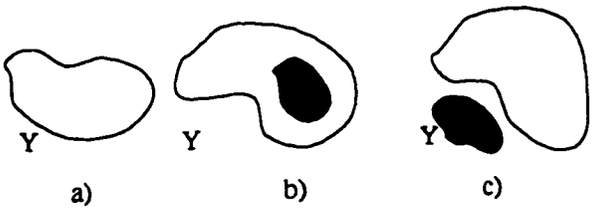

Figure 3.7: The three possible inclusions relations of Y.

$$X_{h_{min}} = T_{h_{min}}(I) \tag{3.21}$$

$X_{h_{min}}$ is made of the points of I which belong to the minima of lowest altitude.

Consider now the threshold of I at level $h_{min}+1$, i.e. $T_{h_{min}+1}(I)$. Obviously,

$$X_{h_{min}} \subseteq T_{h_{min}+1}(I) \tag{3.22}$$

Y being one of the connected components of $T_{h_{min}+1}(I)$, there are three possible relations of inclusion between Y and $Y \cap X_{h_{min}}$:

a) $Y \cap X_{h_{min}} = \emptyset$ — in this case, Y is obviously a new minimum of I. Indeed, according to the definitions in the previous section, Y is a plateau at level $h_{min}=1$, since

$$\forall p \in Y, \begin{cases} p \notin X_{h_{min}} & \Rightarrow \quad I(p) \geq h_{min}+1 \\ p \in Y & \Rightarrow \quad I(p) \leq h_{min}+1 \end{cases} \tag{3.23}$$

Moreover, all the surrounding pixels do not belong to $T_{h_{min}+1}(I)$ and have therefore a graylevel strictly greater than $h_{min}+1$. The minimum thus

discovered is *pierced*, hence its corresponding catchment basin will be progressively filled with water.

b) $Y \cap X_{h_{min}} \neq \emptyset$ **and is connected** — in this case, Y corresponds exactly to the pixels belonging to the catchment basin associated with the minimum $Y \cap X_{h_{min}}$ and having a gray level lower or equal to $h_{min} + 1$:

$$y = C_{h_{min+1}}(Y \cap X_{h_{min}}) \tag{3.24}$$

c) $Y \cap X_{h_{min}} \neq \emptyset$ **and is not connected** — Y contains different minima of I. Denote $Z_1, Z_2, \ldots, Z_k$ these minima, and let Z_i be one of them. At this point, the best possible choice for $C_{h_{min}+1}(Z_i)$ is given by the geodesic influence zone of Z_i inside Y:

$$C_{hmin+1}(Z_i) = iz_Y(Z_i) \tag{3.25}$$

These inclusion relationships are illustrated in Figure 3.7. Since all the possibilities have been discussed, the second set of the recursion is the following one:

$$X_{h_{min}+1} = min_{h_{min}+1} \cup IZ_{T_{h_{min}+1}(I)}(X_{h_{min}}). \tag{3.26}$$

This relation holds for all levels h, and finally, the following definition is obtained:

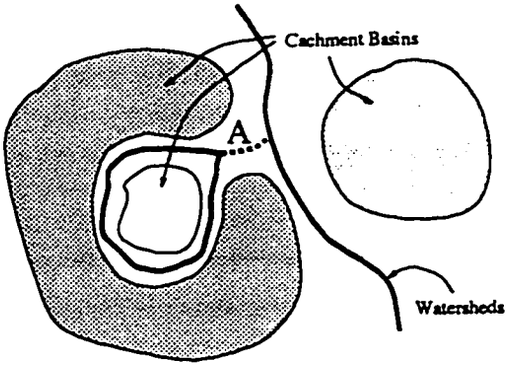

Figure 3.8: Edge A does not belong to the set of watershed lines.

Definition 3.9 (Catchment basins and watersheds by immersion) *The set of the catchment basins of the grayscale image I is equal to the set $X_{h_{max}}$ obtained after the following recursion:*

$$X_{h_{min}} = T_{h_{min}}(I)$$

$$\forall h \in [h_{min}, h_{max} - 1], X_{h+1} = min_{h+1} \cup IZ_{T_{h+1}(I)} \cdot (X_h) \tag{3.27}$$

The watersheds of I correspond to the complement of this set in D_I — i.e. to the set of the points of D_I which do not belong to any catchment basin.

3.4.3 Review of Existing Watershed Algorithms

Approaches to linear feature extraction using mathematical morphology have been reviewed in Chapter 2. Beucher and Lantuéjoul [9] proposed a watershed algorithm based on immersion. In this algorithm, geodesic influence zones are grown

via binary thickenings until idempotence. This algorithm can compute watersheds which, in some special configurations, may contain undesirable arcs as illustrated in Figure 3.8. Arc A is not a watershed line since it does not separate two different minima. Furthermore, the whole set of pixels needs to be scanned at each thickening step, which is time consuming.

Friedlander [21] introduced a sequential watershed algorithm which scans the image in a predefined order, in which the new value of each pixel is immediately taken into account for the computation of the next pixel values. An initial propagation step yields a *broad catchment basin*. A broad catchment basin associated with a minimum M is the set of pixels that can be reached by following a never descending path starting from M. Every pixel in the image belongs at least to one broad catchment basin. The zone where two or more broad catchment basins overlap is called a *watershed zone*. Its complement constitutes the *restricted catchment basin*. Since labeling of catchment basins is used, such traps as that of Figure 3.8 are avoided. However, watershed lines may be improperly located due to the propagation step being based on the definition in terms of flow paths.

3.4.4 Parallel Watershed Transformation

The method described in this section introduces an efficient computation of geodesic influence zones through the use of mathematical morphology on a SIMD parallel architecture.

General Description

The proposed algorithm is based on the definitions given in section 3.4.2. Therefore, one has to consider the successive thresholds of the image under study, and to compute geodesic influence zones of one threshold inside the next. The parallel algorithm is thus centered around the progressive flooding of the catchment basins of the image.

Due to the SIMD parallel architecture, immediate access is available to the pixels at a given level h through binary thresholding. Thus, the initial sorting of the pixels in the increasing order of their levels h is made redundant.

Suppose the flooding has been done up to a given level h. Every catchment basin already discovered — i.e., every catchment basin whose corresponding minimum has an altitude lower or equal to h — is supposed to have an unique label. Thanks to the parallel architecture, the set of pixels having level $h + 1$ are accessed directly. At this stage these pixels can assume one of three roles, as illustrated in Figure 3.7:

a) they can have no labeled pixels among their close neighbors (see equation 3.23) — these pixels represent new *minima* at level $h + 1$.

b) they can have labeled pixels among their close neighbors where the labeled pixels have the same label (see equation 3.24) — i.e. all these labeled pixels belong to the same geodesic influence zone.

c) they can have labeled pixels among their close neighbors where the labeled pixels

have two or more different labels (see equation 3.25) — i.e. these labeled pixels belong to at least two geodesic influence zones.

To properly label these pixels, we perform a two step process. First, we label the pixels which have labeled pixels among their neighbors. Thus we extend the labeled catchment basins by performing successive morphological openings. Where two or more different catchment basins meet, new watershed points are identified. At the end of this step, all new watershed points at level $h + 1$ have been found. After this step, only the new *minima* at level $h + 1$ have not been reached. Indeed, they are not connected to any of the already labeled catchment basins. In the second step, the remaining pixels are given new distinct labels to be thus newly discovered catchment basins.

The parallel watershed algorithm solves some of the problems found in existing algorithms reviewed in section 3.4.3. First, the labeling of the catchment basins automatically avoids such traps as that of Figure 3.8. Now, in order to get perfectly located watershed arcs, the successive geodesic SKIZ involved in the process have to be as good as possible. The first thing to notice is that, according to the discrete distance associated with the underlying grid, the set of pixels which are equidistant to two given connected components may well not be a line, but a very thick area. Recall that the distance between two pixels is equal to the minimum number of grid edges to cross to go from one to the other. Consequently, some simplistic rules in the computation of the geodesic SKIZs could result in unwanted thick watershed areas. Similarly, suppose that the plateaus at elevation h are currently being flooded, and

0	1	2
7	P	3
6	5	4

Figure 3.9: Location of the close neighbors 0 to 7 of pixel P.

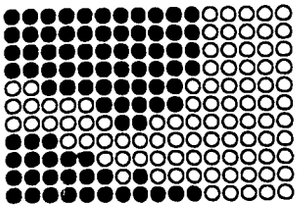

Figure 3.10: An area of the image with two catchment basins $C(B_1)$ and $C(B_2)$ shown in black.

let p be the current pixel. A simple rule would be to say that p is necessarily a watershed pixel (i.e. belongs to the set of watershed lines) if it has a watershed pixel in its neighbourhood. This could result in deviated watershed lines.

The parallel watershed algorithm defines a watershed pixel as being any pixel which has at least two pixels with different labels in its neighbourhood. This approach eliminates the deviated watershed line problem.

The idea is to successively grow the catchment basins in conjunction with carefully written rules for the propagation of the labels inside the plateaus. which results in very well located watershed lines. even in the case of minima embedded in large plateaus.

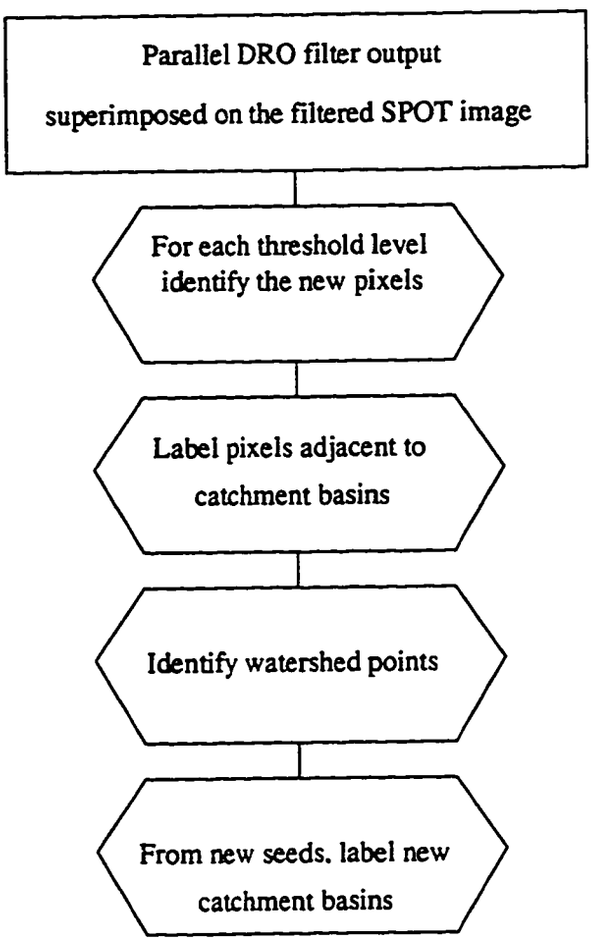

Figure 3.11: The parallel watershed algorithm.

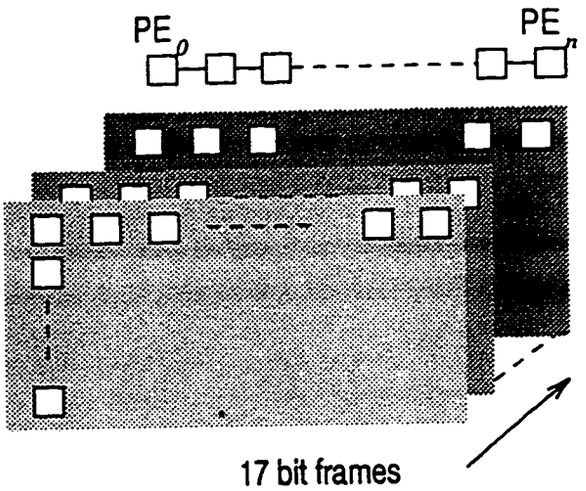

Figure 3.12: The 17 bit im_{label} label frame.

3.4.5 Parallel Watershed Algorithm

The following section introduces the parallel watershed algorithm. The parallel watershed contains four parallel blocks, which follow each other sequentially. Figure 3.11 illustrates the block organization of this algorithm, while figure 3.9 illustrates the location of the eight close neighbors of a pixel. At any time during the watershed algorithm, the image contains a number of Catchment Basins $C(B_i)$. Figure 3.10 shows two such catchment basins, $C(B_1)$ and $C(B_2)$ respectively, separated by several points which have not yet been labeled. The two catchment basins encompass two image areas which have already been labeled as $label_1$ and $label_2$ respectively.

At the beginning of the algorithm, a 17 bit *label* image is created which has a different grayscale value at each pixel coordinate. This label image is used to label pixels at each level h in parallel. Figure 3.12 illustrates the 17 bit *label* image.

The parallel watershed algorithm is introduced below.

Algorithm

- INPUT/OUTPUT:

  input: im_i, SPOT image with the results of the parallel DRO filter superimposed;

  output: im_o, binary image of the watersheds;

  label frame: im_{label}, the 17-bit label frame;

- INITIALIZE 17 BIT LABEL FRAME:

  — A unique value is assigned to each pixel in the label frame im_{label}:

$$\forall p, p' \in D_{im_{label}}, im_{label}(p) = n, \text{ such that, } im_{label}(p) \neq im_{label}(p') \quad (3.28)$$

  — Value 0 is assigned to each pixel of im_o:

$$\forall p \in D_{im_o}, im_o(p) = 0 \quad (3.29)$$

  — Add the value δ to the grayscale value $im_i(p)$ of each pixel p in the input image im_i.

  Since the parallel watershed algorithm attempts to identify all the roads in the image — i.e. bright linear features, increasing the brightness of the image by the use of the linear filter:

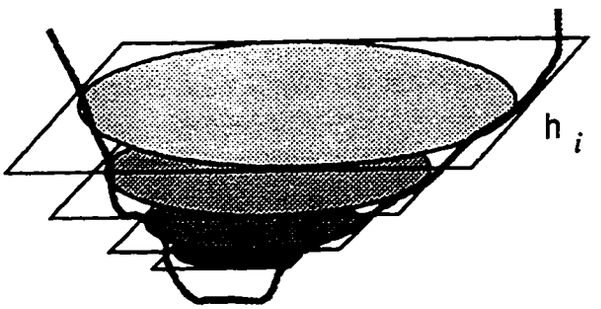

Figure 3.13: Successive threshold operations on a typical catchment basin.

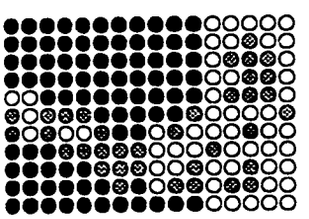

Figure 3.14: The new points $T_h(I)$ (shown in gray). after a threshold operation.

$$im_i(p) = im_i(p) + \delta \tag{3.30}$$

has the effect of strengthening the bright features in the grayscale image im_i before the watershed algorithm begins.

- Let h_{min} and h_{max} designate the lowest and highest graylevel values in the input image, respectively. Figure 3.13 illustrates successive threshold operations on a typical catchment basin.

- For $h \leftarrow h_{min}$ to h_{max} DO:

  — Identify the set of points $T_h(I)$. Figure 3.14 shows two labeled catchment basins $C(B_1)$ and $C(B_2)$. shown in black. The new points $T_h(I)$ are shown

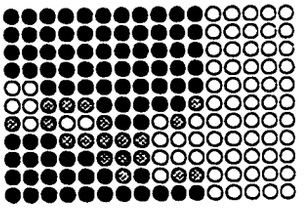

Figure 3.15: The points $U_h(I)$ (shown in gray), which are adjacent to existing *catchment basins* $C(B_1)$ and $C(B_2)$.

in gray. A subset of these points, denoted as $U_h(I)$, represent the points which are adjacent to existing catchment basins, as shown in Figure 3.15.

$$U_h(I) \subset T_h(I) \tag{3.31}$$

At this point, the new points identified in $T_h(I)$ are either unique points — i.e. no labeled neighbors, or non-unique points — i.e. they have neighbors, as shown in Figure 3.14. They are either:

(i). adjacent to *catchment basins* — i.e. they have labeled pixels among their close neighbors.

(ii). adjacent to new points in $T_h(I)$ — i.e. they have non-labeled new points among their close neighbors.

(iii). not adjacent to any pixels, in which case these are new seeds or *minima*.

Based on the above classification, a two step process will be introduced:

— identify all points from $U_h(I)$ that have neighbors and label. This operation uses the SIMD architecture to execute the labeling of all new pixels

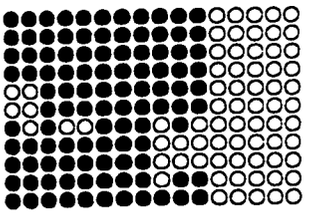

Figure 3.16: Identify the watershed points (shown in dark gray), from the points adjacent to existing *catchment basins*. The adjacent points are labeled with respective *catchment basins* labels.

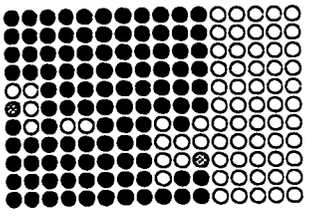

Figure 3.17: New adjacent points to existing *catchment basins*.

in parallel. Select watershed points amongst these. This is a parallel recursive process.

— identify all unique points and label with new unique labels. These are the seeds for new catchment basins.

- WHILE new points exist in $T_h(I)$ DO:

  — Conditional dilate once the existing $C(B_i)$ points using the new points in

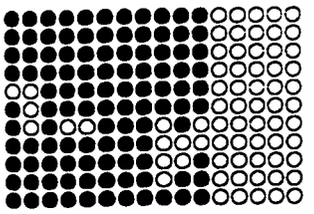

Figure 3.18: Label the new points with the respective *catchment basins* labels.

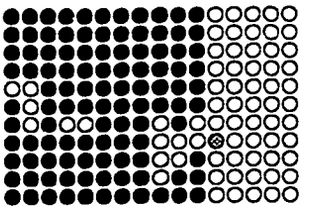

Figure 3.19: New adjacent points to existing *catchment basins.*

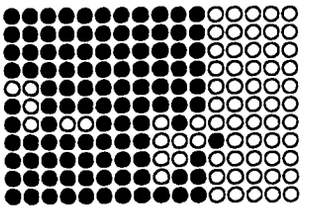

Figure 3.20: Label the new points with the respective *catchment basins* labels.

$T_h(I)$ as the conditional image and label.

— Separate these newly labeled points in $U_h(I)$, as shown in Figure 3.15.

- WHILE new points exist in $U_h(I)$ DO:

  — Identify the watershed points amongst these dilate points using the fact that each new dilate point has a label which has the following properties:

  1. if the point is adjacent to only one $C(B_i)$ then it's new label corresponds to the label of the catchment basin.
  2. if the point is adjacent to more than one $C(B_i)$ then it's new label has the following property:
     - ◇ the label corresponds to the label of one and only one of the catchment basins, or
     - ◇ the label is different from the labels of the adjacent catchment basins.

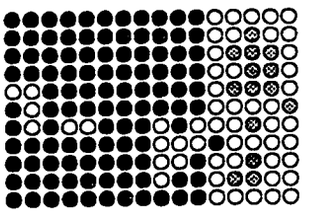

Figure 3.21: New points (shown in gray), which are not adjacent to *catchment basins.*

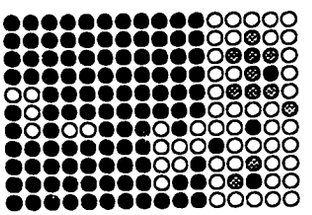

Figure 3.22: Identify the seeds of these new points (shown in black).

Figure 3.16 through Figure 3.20 show the iterative process by which watershed point candidates are labeled depending on their proximity to labeled pixels in catchment basins.

— To identify if a point is a watershed point we must exploit the above properties. Thus we compare the label of each point to its eight close neighbors and identify if this point has a label which is similar to any adjacent $C(B_i)$ label. If a difference occurs we mark this point as a watershed point.

— Remove these points from $T_h(I)$ and $U_h(I)$.

- ENDWHILE $U_h(I)$
- ENDWHILE $T_h(I)$
- IF points left outside dilate points, THEN:

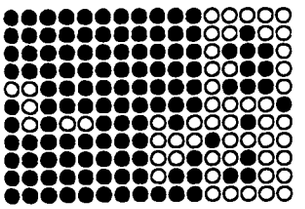

Figure 3.23: Label these new points with respective new *catchment basin* labels $C(B_3), C(B_4)$, and $C(B_5)$ respectively.

— This operation identifies any remaining points found in $T_h(I)$ which fall in category (iii) above

— Identify all the new seeds from the remaining points, as shown in Figure 3.21.

— Label these new seeds, as shown in Figure 3.22

— Label the neighbors of the new seeds also and add these new catchment basins to the set $C(B_i)$ as shown in Figure 3.23

- ENDFOR

The parallel watershed algorithm implemented in Intelligent C is shown in Appendix A.

3.4.6 Initial Conditions

The parallel watershed algorithm receives as input the filtered image with the marking function output superimposed. The marking function restricts the input

domain by making a prediction relative to the location of the features of interest in the image.

3.4.7 Watershed Transformation Stability

The parallel watershed transformation is invariant to scaling, translation, and rotation. The watershed is sensitive to background noise. This characteristic may result in over segmentation.

A discussion of the parallel watershed stability is presented in Chapter 5.

3.5 Conclusion

In this chapter we introduced the parallel algorithms which are at the heart of the linear feature extraction system. First we described the DRO filter used as a marking function. Second, we described the parallel watershed transformation which takes as input the marking function result and proceeds to obtain an image tesselation based on an immersion technique. The parallel watershed algorithm avoids some well known traps which may have an adverse effect on the results.

In the next chapter, the implementation of the linear feature extraction system is introduced. This system takes the algorithms described above and combines them in an end-to-end road network detection system.

Chapter 4

An Object Oriented Environment for Linear Feature Extraction

In the previous chapter, the parallel algorithms used in the design of the road detection system have been introduced. In this chapter, we concentrate on the implementation of the linear feature extraction system with application to the detection of road networks in aerial images. We first give an introduction of the AIS-3500 massively parallel SIMD vision computer and the object oriented environment. Subsequently, the system implementation is described and the general system algorithms are shown. Further discussion of each building block follows with emphasis on the various techniques used which lead to the successful extraction of the linear features of interest. Complete listings in Intelligent C of all algorithms used in the recognition process are given in Appendix A. Appendix B introduces the object model for the GeoFind system. Appendix C describes the available functionality

through simple examples.

4.1 The AIS-3500 Vision Computer

The parallel algorithms described in the previous chapter have been implemented in the Applied Intelligent Systems Inc.(AISI) language, Intelligent C, which uses the LAYERS Object Library as an object oriented extension to the C language [4].

4.1.1 The Architecture of the AIS-3500

The Applied Intelligent Systems Inc. (AISI) AIS-3500 is a massively parallel vision computer with up to 1024 processing elements arranged in a single instruction multiple data (SIMD) architecture. The processing elements (PE) are arranged in a one dimensional chain which can be as wide as the image itself. The PEs take the image column by column and process it in parallel. The parallelism inherent in the architecture must be taken into account when developing image processing algorithms. The AISI-3500 can be programmed in C or the Intelligent C object oriented language. The system provides an extensive software library which includes a C Layer [2] and an Object Layer [4]. The Object Layer includes feature extraction objects, shape objects, and interface objects.

The individual processing element of the AIS-3500 is a general purpose bit se-

rial processor that can perform three types of operations: boolean, arithmetic, and neighbourhood. Boolean operations are characterized by the number (two, three, or four) of input variables, where the source inputs are bits read from parallel memory. The inputs are transformed according to a truth table. In the neighbourhood operations, a logical operation is performed upon a set of bits to produce a unary result. In arithmetic operations, a sum bit and a carry bit are generated for two source bits and a carry bit input.

The architecture of the AIS-3500 makes neighbourhood and morphological operations the most efficient vision operations used when developing image processing applications. This truism is extremely powerful in all the image processing steps developed for the linear feature extraction system which include preprocessing, filtering, segmentation, and post-processing.

4.2 Development Environment on the AIS-3500

In this section, the LAYERS development environment used in applications development on the AIS-3500 is introduced [3].

LAYERS is a set of programming tools for creating image processing applications that run on AISI's vision computers. LAYERS supports application development in C and Intelligent C. It contains:

◇ a library of C functions [2] and Intelligent C classes [4] for system and hard-

ware control, image processing, feature extraction, and graphical user interface (GUI) development

- ◇ utilities for building image processing applications on a host computer (i.e. a UNIX workstation or a PC) and downloading them to run on AISI parallel machines

- ◇ a terminal emulation program which connects a host computer to the parallel machine through a serial communication (RS-232) port and allows the user to transfer image applications (i.e. compiled executable code). It acts as the programmer interface to the vision computer, and can transfer data files between the host computer (UNIX workstation or PC) and the vision computer (AIS-3500).

4.3 The GeoFind Object Oriented Environment

The road extraction system has been developed using the LAYERS development environment and Intelligent C. The object oriented approach in the design and implementation of the linear feature extraction system offers advantages inherent in object oriented design [14], as well as access to the library of image processing classes provided by the AISI Object Layer [4].

The linear feature extraction system is called *GeoFind* version *2.4* and it provides a unified end-to-end platform for the automated linear feature extraction mechanism.

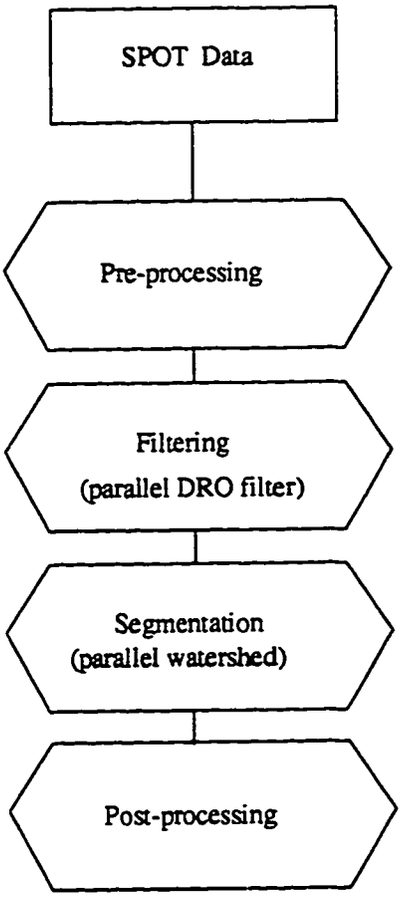

Figure 4.1: The road detection algorithm.

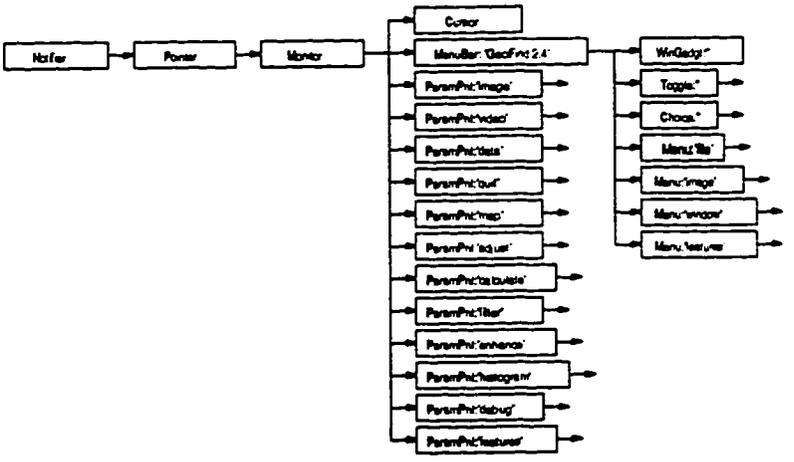

Figure 4.2: GeoFind graphical object tree showing the top level only.

4.3.1 GeoFind Object Model

The Geofind system has been developed based on an object model illustrated in Figure 4.2 (top level only). Please refer to Appendix B for a description of the complete object model. The recognition system encompasses the following steps as illustrated in Figure 4.1:

pre-processing The original aerial image is inverted if necessary to ensure that the features of interest are lighter than the background. Brightness and contrast adjustment is performed. If more than one channel is available. addition and subtraction between channels can be performed.

filtering The parallel DRO algorithm is used to create the marking function.

segmentation The parallel watershed algorithm is used to extract the linear features.

post-processing Contextual information is used to eliminate superfluous lines and to connect road segments.

The next sections describe in detail the implementation steps outlined above. Relevant areas of the GeoFind system are illustrated. Please refer to Appendix C for a description of the functionality of the GeoFind system.

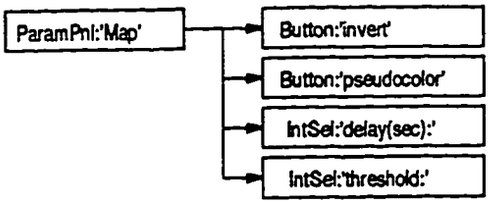

Figure 4.3: Graphical object tree for the *Map* panel interface object.

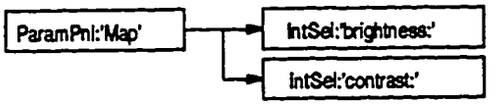

Figure 4.4: Graphical object tree for the *Adjust* panel interface object.

4.4 Image Pre-Processing

The first step in the recognition process is the *pre-processing* step. In this step, the original image is filtered to accentuate the features of interest. The available operations are:

invert This operation allows for the inversion of the grayscale image. This step is necessary, since we must ensure that the road network is lighter than the background for the watershed transformation to perform successfully. Please refer to Figure 4.3 for an illustration of the graphical object tree of the *Map* panel interface object which includes the *invert* method.

The parallel implementation of the *inversion* operation in Intelligent C is given in section A.2.

brightness The brightness of the image can be varied to ensure that the features of interest are displayed taking advantage of the full grayscale spectrum. Quite

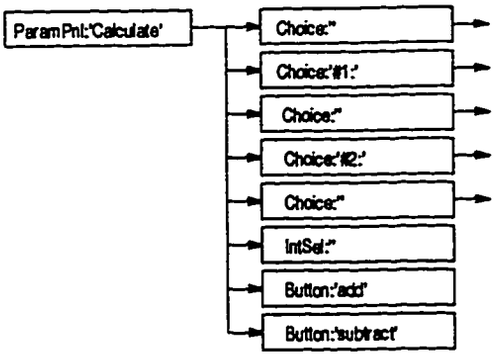

Figure 4.5: Graphical object tree for the *Calculate* panel interface object.

often, aerial images appear very faint with a high degree of background noise. This function enhances the visibility of the elongated features. Please refer to Figure 4.4 for an illustration of the graphical object tree of the *Adjust* panel interface object which includes the *brightness* method.

contrast The contrast of the image can be finetunned to ensure maximum feature separation. Please refer to Figure 4.4 for an illustration of the graphical object tree of the *Adjust* panel interface object which includes the *contrast* method.

The brightness w and contrast b parameters are used to transform the original image as follows:

$$y = 255/(w-b) \times x - 255 \times b/(w-b) \tag{4.1}$$

The parallel implementation of the *brightness* and *contrast* operations in Intelligent C is given in section A.2.

add/subtract This function allows for up to six different channels of an aerial image to be added or subtracted. The user may combine several channels which

exhibit good separation of the features of interest. Please refer to Figure 4.5 for an illustration of the graphical object tree of the *Calculate* panel interface object which includes the *add* and *subtract* methods.

The parallel implementation of the *add* operation is shown below:

- Add or subtract a constant value from the source channel.
- Add the two channels.
- Add or subtract a constant value from the result image.

The parallel implementation of the *add* operation in Intelligent C is given in section A.2.

The parallel implementation of the *subtract* operation is shown below:

- Add or subtract a constant value from the source channel.
- Subtract the two channels.
- Add or subtract a constant value from the result image.

The parallel implementation of the *subtract* operation in Intelligent C is given in section A.2.

4.5 Image Filtering

The next step in the recognition process is the filtering operation. The marking function calculated in this step is used to identify the approximate location of

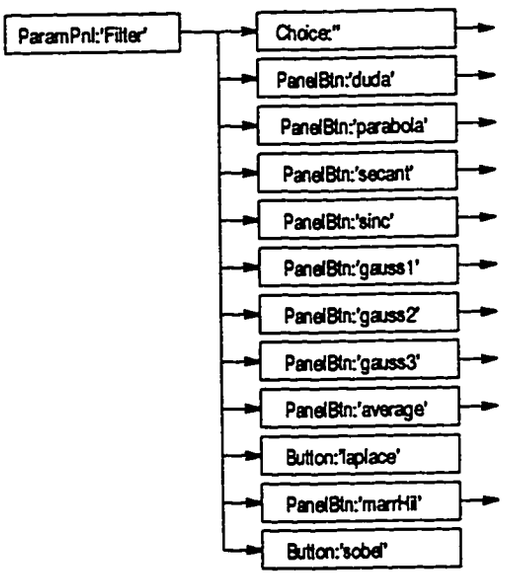

Figure 4.6: Graphical object tree for the *Filter* panel interface object.

the elongated features. The result of this filter operation will be used as the the initial condition for the watershed transformation to follow. The following filters are available for the calculation of the marking function:

- ◇ Parallel Duda Road Operator (DRO) filter
- ◇ Parabola filter
- ◇ Secant filter
- ◇ Sinc filter
- ◇ Gauss filter (one peak)
- ◇ Gauss filter (two peaks)
- ◇ Gauss filter (mathematical morphology)

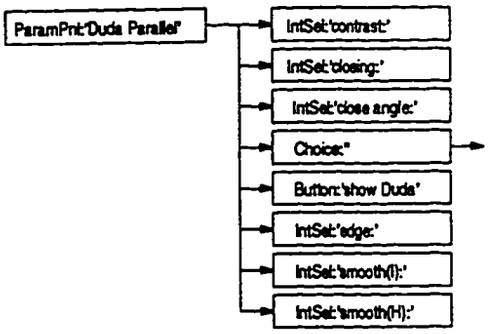

Figure 4.7: Graphical object tree for the *Duda Road Operator* panel interface object.

◇ Average filter

◇ Laplace filter

◇ Marr-Hilldred filter

Please refer to Figure 4.6 for an illustration of the graphical object tree of the *Filter* panel interface object which includes the *filter* methods.

Tests performed with all the above filters have identified the parallel DRO filter as the most likely candidate for the marking function. Results of the application of these filters on aerial images are shown in Chapter 5.

From Chapter 3, we recall that the parallel DRO filter uses a scoring function $SCORE$, given in equation 3.2, for the detection of linear features. This function is calculated for each of the four masks (horizontal, vertical, right diagonal, and left diagonal), with the maximum value $SCORE_{max}$ replacing the intensity value of a given pixel a_2 in the center of the mask.

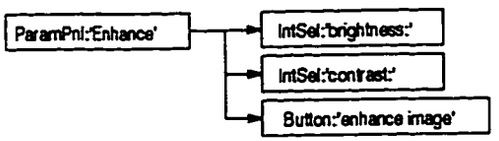

Figure 4.8: Graphical object tree for the *Enhance* panel interface object.

Please refer to Figure 4.7 for an illustration of the graphical object tree of the *Duda Road Operator* panel interface object which includes the methods responsible for calculating the parallel DRO filter output.

The parallel implementation of the parallel DRO filter (see Figure 3.3) is shown below:

◇ Create the look-up tables for $F(u)$ and $G(u)$.

◇ Calculate the value of the $SCORE_H$ function in the horizontal direction.

◇ Once the $SCORE_H$ function has been calculated, we normalize the resulting image by using the formula:

$$y = 255/dudaMax \times i \tag{4.2}$$

◇ The last step involves thresholding the parallel DRO filter output.

- First, we enhance the parallel DRO filter output in anticipation of the thresholding operation by using the brightness and contrast parameters again. These parameters were used previously in the pre-processing stage. Please refer to Figure 4.8 for an illustration of the graphical object tree

of the *Enhance* panel interface object which includes the *brightness* and *contrast* methods.

- Second, we threshold the result image using an parallel adaptive thresholding algorithm.

Finally, we use contextual information to improve the results of the parallel DRO filter in the following areas:

- Reduce thick lines.
- Eliminate dots.
- Perform morphological closings in the *horizontal* direction.
- Eliminate small thin lines.

◇ Similarly, the above procedure is applied in the other directions. At the end of this step, the grayscale of each pixel is replaced with $SCORE_{max}$ given by:

$$SCORE_{max}[A](x) = max\{SCORE \mid (x, SCORE) \in A\} \qquad (4.3)$$

The parallel implementation of the *parallel DRO* filter in Intelligent C is given in section A.3.

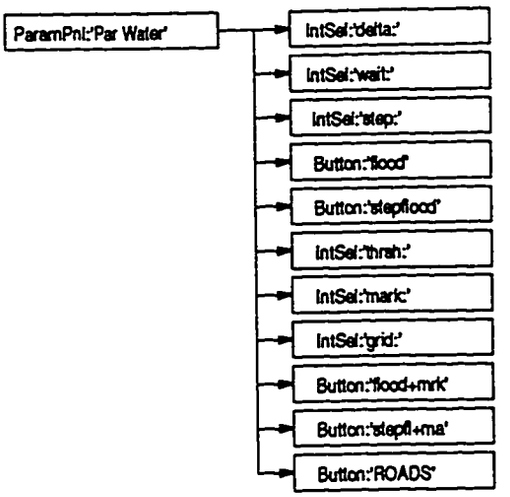

Figure 4.9: Graphical object tree for the *Parallel Watershed* panel interface object.

4.6 Feature Extraction

The next step in the recognition process involves the extraction of the features of interest (roads) from the image. The algorithm developed for this task is the parallel watershed transformation. The starting point for the transformation is the filtered image from the previous step.

From Chapter 3, we recall that the parallel watershed algorithm is based on the definitions given in section 3.4.2. Therefore, one has to consider the successive thresholds of the image under study, and to compute geodesic influence zones of one threshold inside the next. The parallel algorithm is thus centered around the progressive flooding of the catchment basins of the image.

Please refer to Figure 4.9 for an illustration of the graphical object tree of the *Parallel Watershed* panel interface object which includes the methods responsible

for calculating the parallel watershed transformation.

The parallel implementation of the parallel watershed (see Figure 3.11) is shown below:

◇ Increase the contrast of the input and create the label image.

- Create the 17 bit label image which is used to label new pixels at threshold level h in parallel. The depth of this label image is such that it allows unique labeling of pixels in a 512×256 image.
- Add the value δ to the grayscale value $im_i(p)$ of each pixel p in the input image im_i.

  Since the parallel watershed algorithm attempts to identify all the roads in the image — i.e. bright linear features, increasing the brightness of the image by the use of the linear filter:

$$im_i(p) = im_i(p) + \delta \tag{4.4}$$

  has the effect of strengthening the bright features in the grayscale image im_i before the watershed algorithm begins.

◇ Suppose the flooding has been done up to a given level h. Every catchment basin already discovered — i.e., every catchment basin whose corresponding minimum has an altitude lower or equal to h — is supposed to have an unique label. The remaining of this section will show the implementation of the

parallel watershed algorithm for one flooding level. The parallel watershed transformation repeats these steps FOR $h \leftarrow h_{min}$ to h_{max}, in our case $(0, 255)$.

- Add one flooding step *h-1* and get the next flooding step *h*.
  - Identify new pixels.
  - Eliminate the watershed points from the existing GIZs.
- While new points exist DO:
  - Conditional dilate once the existing GIZ points using the new points as the conditional frame.

    Separate these points.

    Dilate once the existing GIZs.

    Separate new points obtained in the dilate and store.

    Complete the dilate operation on the GIZs.
  - The points that are adjacent to existing GIZs are included in their respective GIZs (i.e. they will be labeled respectively) using the 17 bit label image.

    Eliminate existing watershed points from dilated points.

    Label these points with their respective GIZs labels using the 17 bit label image.
  - Identify the new watershed points amongst these dilate points.

    To identify if a point is a watershed point we compare the label of each point to its eight close neighbors and identify if this point has a label which is similar to any adjacent GIZ label. If a difference

occurs we mark this point as a watershed point.

Reduce the dilate points to idempotence.

Identify all the new seeds for the new GIZs.

Identify the labels of the new points.

- Compare the label of point P with the labels of its eight neighbors. If a label of any of the eight neighbors is different, flag the difference. In this case, point P is a watershed point.

  The implementation is shown below for one of the eight neighbors only (neighbor 0). The same algorithm applies to all the eight close neighbors.

  Shift P to position *0*.

  Compare P with label at position *0*.

  Mask out the neighbors with label *0*.

  Mask out neighbors which are watershed points.

  Mask out the result.

  Shift result of compare operation back to position P and store.

- At the end of the above step, points P have been compared with all their eight close neighbors. The flagged points amongst these are watershed points.

  Add these points to the watershed list.

  Eliminate these points from the existing GIZs. This operation is necessary to eliminate the following scenario:

  — a new watershed candidate is labeled by a conditional dilate of

the GIZ label conditional on the T_h footprint. If watershed points are allowed to survive as parts of the GIZs then the labels of these watershed points can corrupt the labeling operation outlined above.

- Remove these new found watershed points from the new points.

  Repeat until no dilate points are left.

  The dilate points have been removed from the new points in T_h.

- Examine if any points have been left in T_h.

  If so, take each point (i.e. seeds), label it, and label its neighbors.

  Identify all the new seeds.

  Test for presence of closed contours in new GIZs:

  While close contours exist:

  Separate this point.

  Add this new seed.

  Conditional dilate this point with the closed polygon in until idempotence.

  Separate the respective closed polygon.

  Check for idempotence.

  Remove the above extracted polygon.

  Label these new seeds.

  Label the neighbors of the new seeds.

  Add these new GIZs to the set.

The parallel implementation of the *parallel DRO* filter in Intelligent C is given in section A.4.

4.7 Image Post-Processing

In this last step in the road detection system, contextual information is being used to connect collinear road segments and to eliminate unwanted small linear features.

Please refer to Figure 4.9 for an illustration of the graphical object tree of the *Parallel Watershed* panel interface object which includes the methods mentioned in this step.

The parallel implementation (see Figure 3.11) is shown below:

- First eliminate small closed contours.
- Next, eliminate small watershed segments.
- Finally, trim existing roads.

The parallel implementation of the *contextual* filter in Intelligent C is given in section A.5.

4.8 Conclusion

In this Chapter, the implementation of the various parallel algorithms used in the detection of the road network from aerial images has been developed.

In the next Chapter, results of the road network extraction system are shown.

Chapter 5

Testing and Results

5.1 Introduction

This chapter discusses the methodology used for the testing of the various filters introduced in Chapter 3. The robustness, performance, and limitations of these algorithms and that of the linear feature extraction system as presented in Chapter 4 are discussed. Further tests show the robustness of the algorithms with respect to small subdivision roads, roads which are partially occluded due to shading or overlap from adjacent vegetation or man made structures, and roads which appear enlarged due to adjacent structures which reflect the same light intensity.

The tests have been done on a set of 512 × 256 pixels, 8 bits, aerial SPOT

images of the urban area of Ottawa-Hull, Canada, provided by the Canadian Centre for Remote Sensing (CCRS). The SPOT High Resolution Visible (HRV) sensor in pan-chromatic, known as PLA, has a spatial resolution of 10 meters. The Canadian Centre for Remote Sensing (CCRS) produces a geocoded digital product that is re-sampled at 6.25 meters. The single band imagery is collected over the 0.51 to 0.73 micrometer spectral region, with a dynamic range of 0 to 255. These images are similar in content with the images used by other researchers in bringing relevance to their approaches in road network detection [11], [49], [20], [22], [34], [44], [53], and [54].

The testing which has been performed in order to validate the approach presented in this thesis covers the following areas:

- a number of various aerial images have been used to verify the performance of the parallel DRO filter. The frequency, location, orientation, and size of the line segments have been varied to demonstrate the robustness of the marking function with respect to these parameters. Examination of the results obtained with the parallel DRO filter are as expected when compared with the sequential DRO filter [44].
- post-processing performed on the parallel DRO filter results has been evaluated paying particular attention to the ability of the algorithm to connect collinear line segments.
- a number of different aerial images have been used to assess the validity of the

Figure 5.1: *IMAGE 1*:SPOT image of a section of downtown Ottawa, ON, Canada

parallel watershed transformation. In particular, the special cases referred to in Chapter 3 have been tested to verify the performance of the parallel implementation. Comparison of the output of the parallel watershed transformation with the result of the sequential watershed [47], [52], [46] is discussed.

- finally, different scenarios have been selected for image post-processing of the road network system results. Here, issues such as, elimination of small catchment basins, elimination of perceived noise (i.e. small line segments and dots), and the performance of the filter with respect to known special cases has been addressed.

5.2 Pre-Processing Phase

The pre-processing phase involves capturing the scenes in the SPOT images and performing enhancement operations to improve the contrast of the road networks. The road network recognition system works on the assumption that the features of

Figure 5.2: *IMAGE 2*:SPOT image of a section of downtown Hull, QC, Canada

Figure 5.3: *IMAGE 3*:SPOT image of a section of a Hull suburb, QC, Canada

Figure 5.4: *IMAGE 4*:SPOT image of a section of Gatineau, QC, Canada

Figure 5.5: *IMAGE 1* with contrast adjustment

Figure 5.6: *IMAGE 2* with contrast adjustment

Figure 5.7: *IMAGE 3* with contrast adjustment

Figure 5.8: *IMAGE 4* with contrast adjustment

interest, in our case the road networks, are lighter than the surrounding background. The enhancement operations available are:

inversion The image can be inverted to ensure that the background is darker than the road network. Figures 5.1 through 5.4 show urban scenes which were selected for the purposes of our tests. Since these images show the roads as light areas, no inversion of the images was necessary.

brightness The brightness of the image can be modified to compensate for aerial images which have a low intensity level. No changes in brightness levels were done for the test images.

contrast The contrast of the image can be modified to improve the separation between features. Figures 5.5 through 5.8 show the original scenes with some of the high frequency noise removed. Here, the contrast is increased to improve the separation between the road networks and the image background.

Figure 5.9: *IMAGE 1* filtered with the parabola filter — Upper Bound = 255, middle = 128, Lower Bound = 0, min at UB = 0, min at LB = 0, max value = 255

Figure 5.10: *IMAGE 1* filtered with the secant filter — max value = 255, middle = 128, relative bell width = 0.5

channel mixing In the case of multiple channels being available for a SPOT scene, this operation allows for the addition or subtraction of different channels for improved feature separation. No channel mixing was done on the test images.

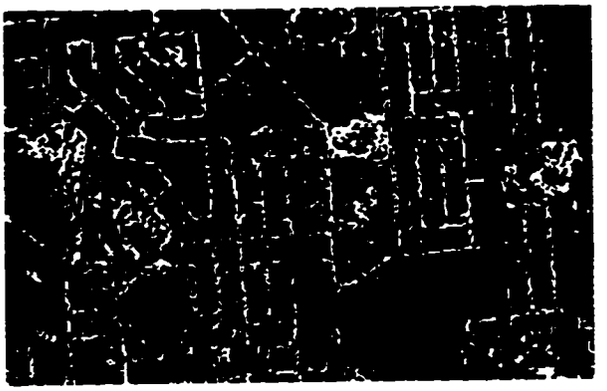

Figure 5.11: *IMAGE 1* filtered with the sinc filter — max value = 255, middle = 128, relative bell width = 0.5

Figure 5.12: *IMAGE 1* filtered with the Gauss filter — max value = 255, middle = 128, relative bell width = 0.5

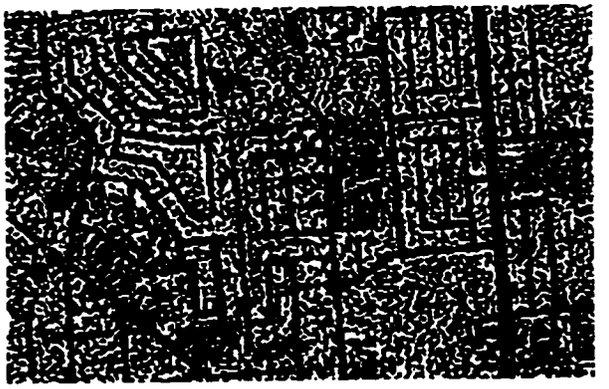

Figure 5.13: *IMAGE 1* filtered with the Laplace filter

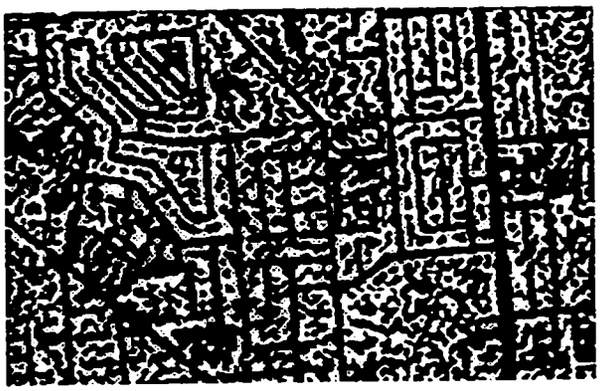

Figure 5.14: *IMAGE 1* filtered with the Marr-Hildreth filter — feature size = 1, feature disp = 2

Figure 5.15: *IMAGE 1* filtered with the sobel filter

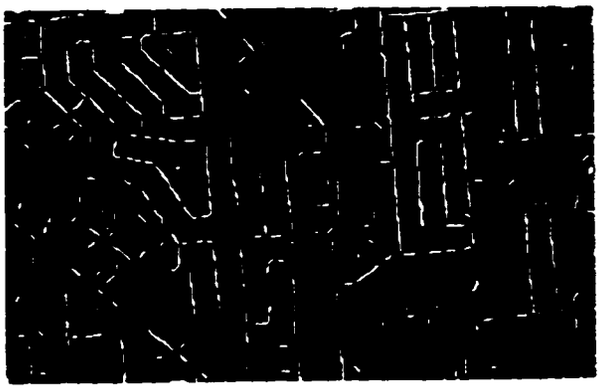

Figure 5.16: *IMAGE 1* filtered with the parallel DRO filter with no post-processing — $\theta = 15$, $\theta_1 = 5$, $\theta_2 = 20$

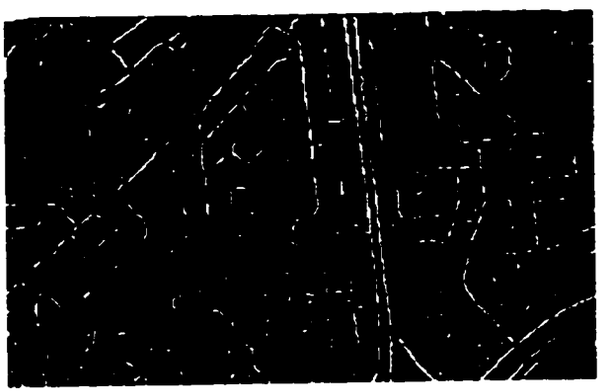

Figure 5.17: *IMAGE 2* filtered with the parallel DRO filter with no post-processing — $\theta = 15, \theta_1 = 5, \theta_2 = 20$

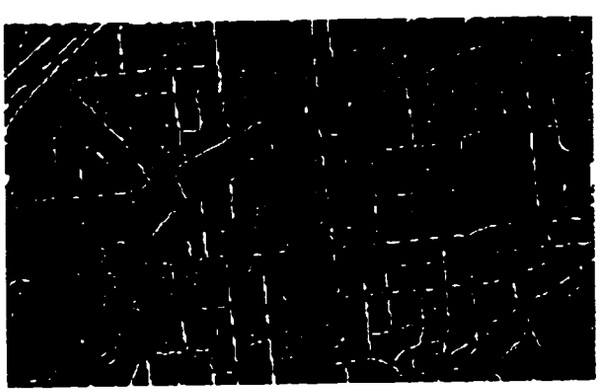

Figure 5.18: *IMAGE 3* filtered with the parallel DRO filter with no post-processing — $\theta = 15, \theta_1 = 5, \theta_2 = 20$

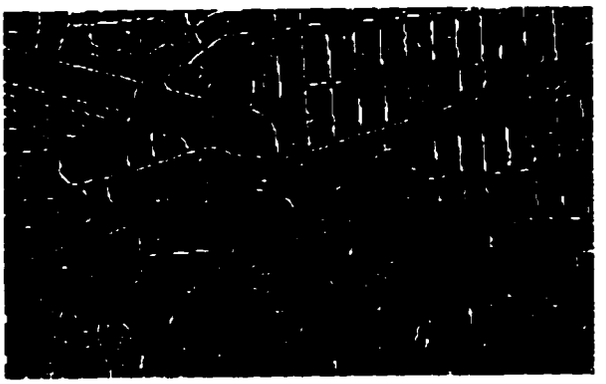

Figure 5.19: *IMAGE 4* filtered with the parallel DRO filter with no post-processing — $\theta = 15$, $\theta_1 = 5$, $\theta_2 = 20$

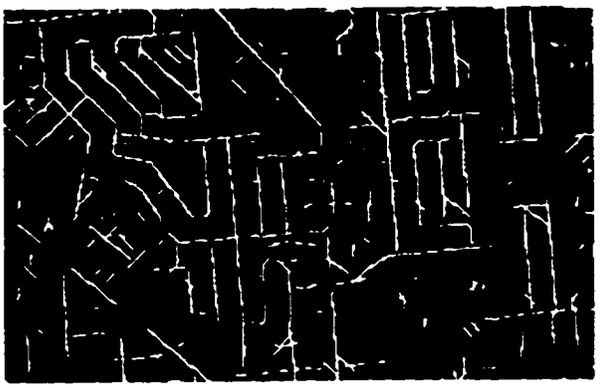

Figure 5.20: *IMAGE 1* filtered with the parallel DRO filter with post-processing — $\theta = 15$, $\theta_1 = 5$, $\theta_2 = 20$

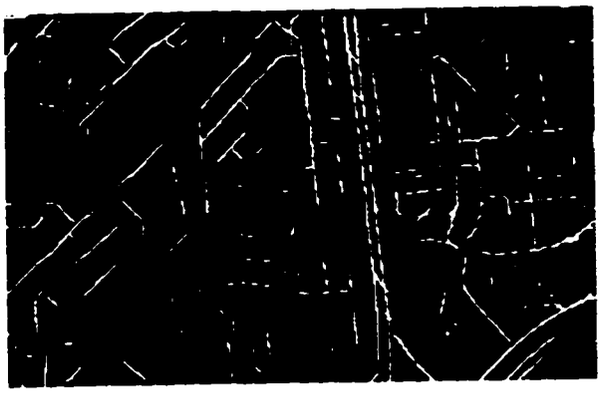

Figure 5.21: *IMAGE 2* filtered with the parallel DRO filter with post-processing — $\theta = 15, \theta_1 = 5, \theta_2 = 20$

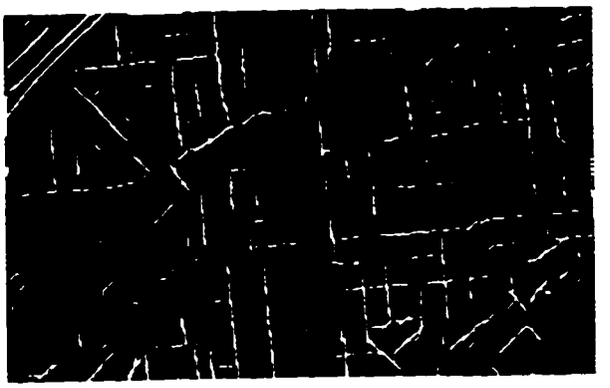

Figure 5.22: *IMAGE 3* filtered with the parallel DRO filter with post-processing — $\theta = 15, \theta_1 = 5, \theta_2 = 20$

Figure 5.23: *IMAGE 4* filtered with the parallel DRO filter with post-processing — $\theta = 15, \theta_1 = 5, \theta_2 = 20$

5.3 Image Filtering

The next step in the recognition process is the filtering of the image. This operation generates the marking function for the next step which is the watershed transformation. Figures 5.9 through 5.15 illustrate several filters which have been considered. The parallel DRO filter exhibits good results when compared with other filters.

The parallel DRO filter uses a scoring function which takes into account contextual information relevant to linear features. The results of the parallel DRO filter without post-processing applied to the urban scene are shown in figures 5.16 through 5.19. Results with post-processing are shown in Figures 5.20 through 5.23, which include elimination of noise and connection of collinear line segments. Clearly, the original parallel DRO result contains a certain degree of noise which is not relevant to the feature recognition task. Once post-processing has been done, most

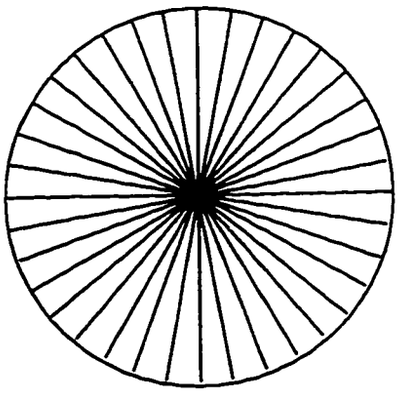

Figure 5.24: Test image used to illustrate the performance of the parallel DRO filter with respect to rotation

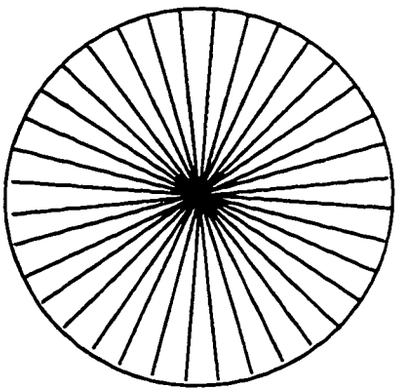

Figure 5.25: Test image rotated by 5 degrees

of the noise has been removed. However, some noise has been erroneously interpreted by the filter to be a valid feature. This error is now introduced in the later stages. As a result, care must be observed in selecting the appropriate levels for the post-processing operation.

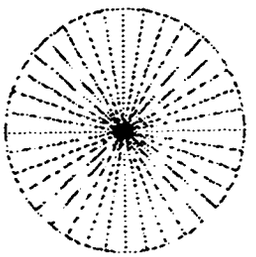

Figure 5.26: Test image bitmap

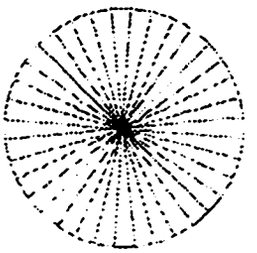

Figure 5.27: Test image bitmap rotated by 5 degrees

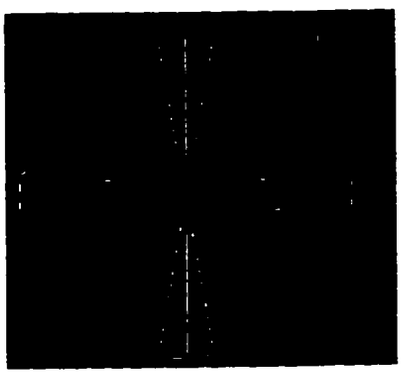

Figure 5.28: Parallel DRO filter result on test image bitmap without post-processing — $\theta = 15, \theta_1 = 5, \theta_2 = 20$

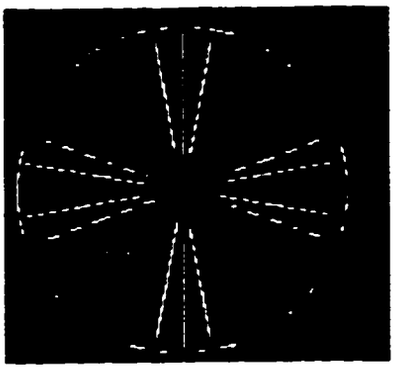

Figure 5.29: Parallel DRO filter result on test image bitmap with post-processing — $\theta = 15, \theta_1 = 5, \theta_2 = 20$

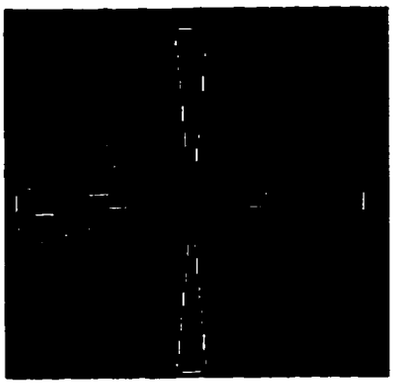

Figure 5.30: Parallel DRO filter result on rotated test image bitmap without post-processing — $\theta = 15, \theta_1 = 5, \theta_2 = 20$

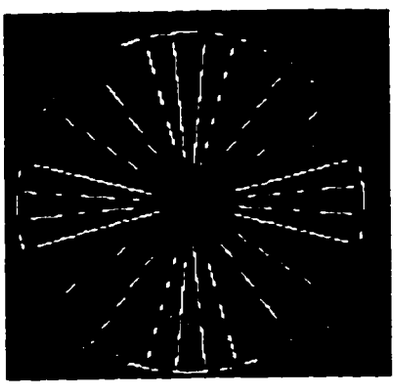

Figure 5.31: Parallel DRO filter result on rotated test image bitmap with post-processing — $\theta = 15, \theta_1 = 5, \theta_2 = 20$

5.3.1 Invariance to Rotation

To verify the stability of the parallel DRO filter with respect to rotation a test image has been used in which lines at intervals of 10 degrees are drawn as shown in figure 5.24. The same test image has been rotated by 5 degrees in figure 5.25. The results of applying the parallel DRO filter to the above test image bitmaps are shown in figures 5.28 through 5.31. These results confirm, that this filter is not invariant to rotation. However, the parallel algorithm shows good results for rotation angles of +/- 5 degrees. These results confirm, as expected, that this filter exhibits a similar response to the sequential DRO filter [44].

5.3.2 Stability of the Duda Filter

The parallel DRO filter uses a scoring function for the detection of straight lines as shown in equation 3.2. This function is applied in conjunction with four

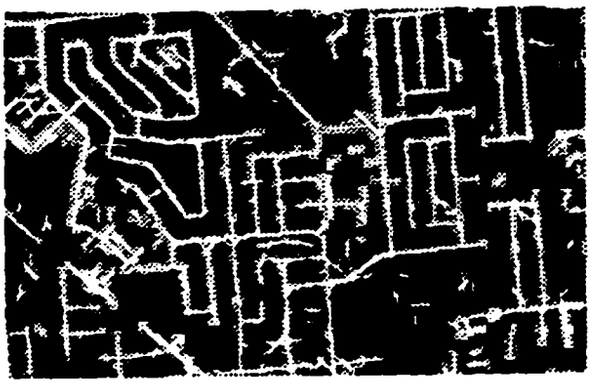

Figure 5.32: The parallel DRO filter as marking function on *IMAGE 1*

different masks identifying four directions: vertical, horizontal, right diagonal, and left diagonal. The three parameters which affect the performance of the filter are:

- θ - linear feature contrast threshold (default value 15) — this parameter controls the minimum edge intensity which will be visible.
- θ_1 - lower intensity threshold along the linear feature (default value 5).
- θ_2 - higher intensity threshold along the linear feature (default value 15) — the above two parameters control the intensity variation along the linear feature.

Wide linear features (e.g. highways) will appear as collinear line segments, with the edge intensity controlled by θ.

Several test were performed with different values for the above parameters and the best results were obtained with the indicated default values. The best performance of the parallel DRO filter is exhibited with the above parameter set values.

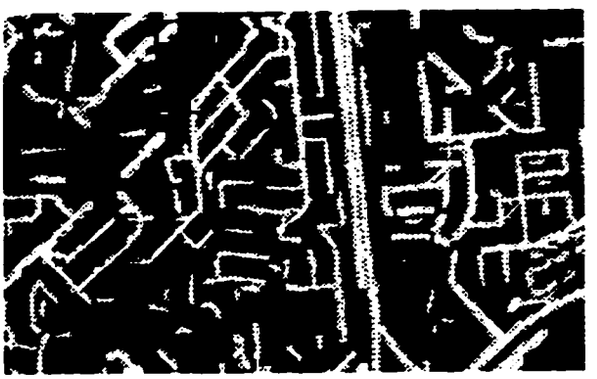

Figure 5.33: The parallel DRO filter as marking function on *IMAGE 2*

Figure 5.34: The parallel DRO filter as marking function on *IMAGE 3*

Figure 5.35: The parallel DRO filter as marking function on *IMAGE 4*

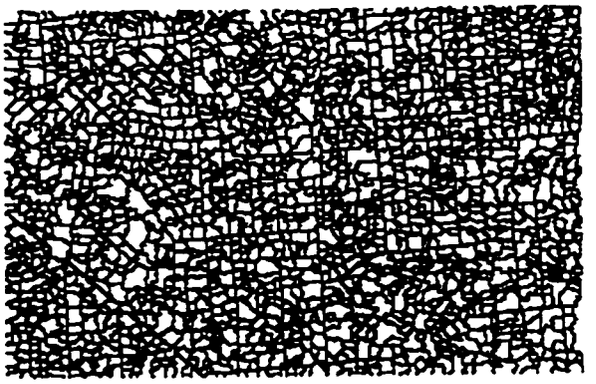

Figure 5.36: The parallel watershed transformation on *IMAGE 1* without a marking function

5.3.3 Parallel DRO Filter as Marking Function

By superimposing the parallel DRO filter output on the original scene, we have highlighted or *marked* the features of interest in anticipation of the watershed transformation.

Before we start the watershed transformation, we therefore *mark* the scene with the parallel DRO filter result. Figures 5.32 through 5.35 show the marking function superimposed on the original scene. Note that the DUDA filter results mark the road candidates, while everything else is shown as background in black.

5.4 Tests on Parallel Watershed Transformation

The road network in the SPOT image is extracted by the parallel watershed transformation by gradually immersing the scene. As the scene is gradually flooded,

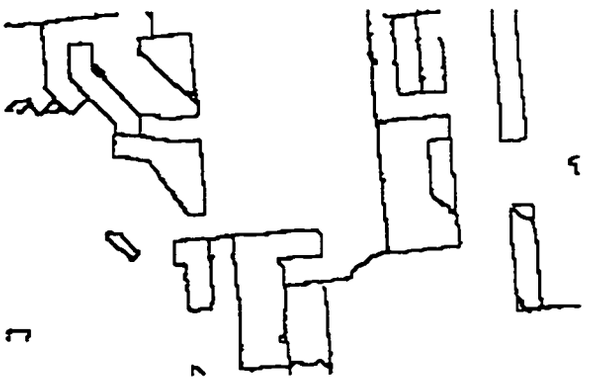

Figure 5.37: The parallel watershed transformation on *IMAGE 1* with the marking function with n=1

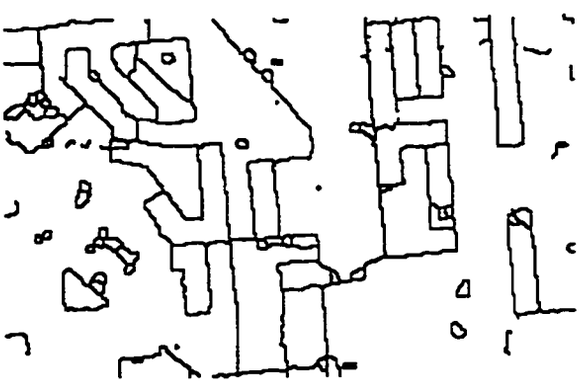

Figure 5.38: The parallel watershed transformation on *IMAGE 1* with the marking function with n=2

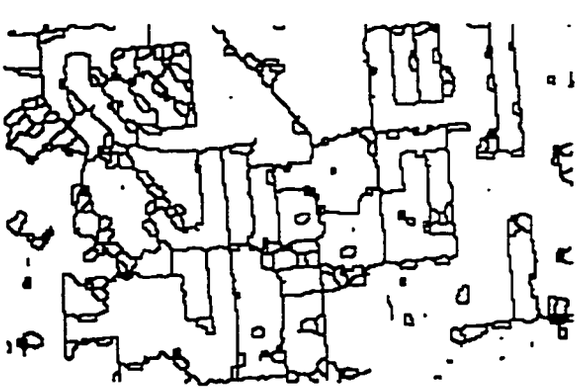

Figure 5.39: The parallel watershed transformation on *IMAGE 1* with the marking function with n=3

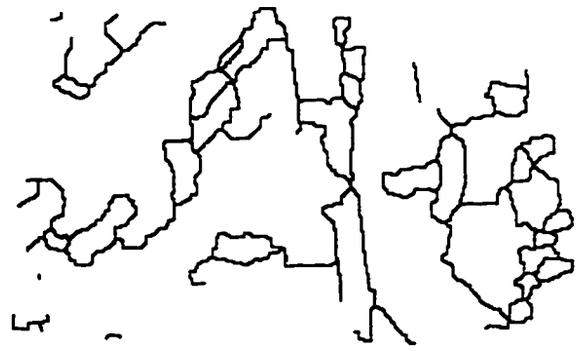

Figure 5.40: The parallel watershed transformation on *IMAGE 2* with the marking function with n=2

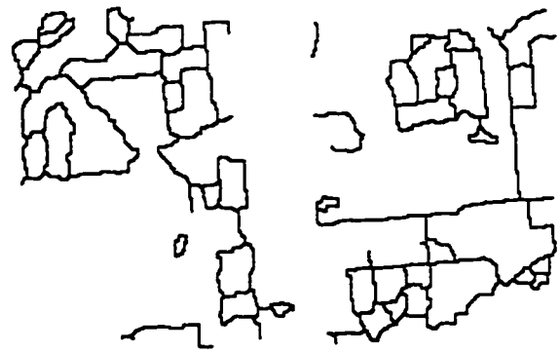

Figure 5.41: The parallel watershed transformation on *IMAGE 3* with the marking function with n=2

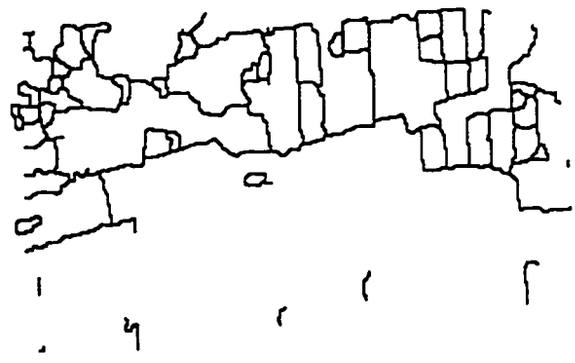

Figure 5.42: The parallel watershed transformation on *IMAGE 4* with the marking function with n=2

the watershed points appear at the points where different catchment basins meet. Since the roads are lighter than the background, the watershed points identify the linear features in the image, or in our case, the road network.

Figure 5.36 shows the result of the parallel watershed transformation on *IMAGE 1* without a marking function. The watershed points are very dense and they cover much more than just the road network due to excessive over segmentation.

Figures 5.37 through 5.39 show the results of the parallel watershed transformation on *IMAGE 1* with the marking function with different post-processing coefficients.

Results of the parallel watershed on different images are shown in Figures 5.40 through 5.42. The results show a drastic reduction in the amount of over segmentation. Note that the parallel watershed algorithm allows open contours which represent strong road candidates.

Comparison of the results with the original images show good levels of road identification. For unbroken linear segments, the results show excellent identification levels of the original roads. Broken linear segments tend to be erased and require careful fine tuning of algorithm parameters (marking function and post-processing) for optimal results.

Brightness and contrast changes in the original aerial images will affect the results. Contrast adjustments during the pre-processing stage are crucial to ensure good results. Light conditions at the time when the aerial image was taken will

greatly affect the quality of the images, and subsequently, the quality of the results. Seasons, cloud cover, time of day, obstructions, and reflectivity of ground, all play an important role in the ability of the recognition system to successfully identify road candidates.

The performance of the road recognition system in identifying road candidates in a 512 × 256 SPOT image varies greatly with scene complexity. The performance of the algorithm is independent of the number of lines and is dependent of the length of the linear features.

5.4.1 Stability of the Watershed Approach

The watershed transformation is highly stable with respect to translation. This is demonstrated through the variety of the linear features present in the images under study. The algorithm is rotation dependent due to the 8 direction (8 connectivity) configuration. This ensures that all linear features are 8 connected.

5.5 Conclusion

The watershed transformation in conjunction with a robust marking function is an excellent method for extracting urban road networks..

Chapter 6

Conclusion

Feature extraction from aerial images continues to generate a great deal of interest in the research community. The different approaches proposed in the past dealt mainly with sequential feature recognition algorithms. This thesis focused on the concept and design of a linear feature extraction system in an object oriented environment, implemented on a massively parallel SIMD architecture. The main thrust of the design was the development of parallel linear feature recognition algorithms using object oriented methodology.

The work discussed in this thesis is a first implementation of a linear feature extraction system with application in road network detection from aerial images. The objectives of this research were:

- design and implement parallel algorithms used in the extraction of linear features from aerial images.

◇ apply this design to the recognition of road networks in aerial images to be integrated into geographic information systems for the purpose of automating the process of updating road maps.

Both these objectives have been met with the implementation of the *GeoFind* system. Results presented in Chapter 5 show that the parallel algorithms developed for this thesis are successful in the extraction of road networks.

The original features of the *GeoFind* system that helped in the fulfillment of the above goals, as well as the contribution of the author towards the solution of this problem are:

- The parallel Duda Road Operator algorithm which is used as the marking function for the linear feature extraction system.
- The parallel watershed algorithm which is used to extract the linear features.
- A set of rules for the implementation of the parallel watersheds which addressed some commonly know limitations.
- The object oriented approach used in the development of the linear feature extraction system is designed to allow future development in this area to build on the existing system.
- The parallel implementation of the end-to-end linear feature extraction system called *GeoFind*.

Future work on this subject will improve the extraction system in the following areas:

- Increase the number of directions used in the calculation of the $SCORE_{max}$ function for the parallel Duda Road Operator from four to eight. This will improve the performance of the algorithm with respect to rotation.
- Improve the parallel DRO algorithm post-processing which uses contextual information to remove small road segments or connect collinear roads by adding the ability to connect road segments which are not collinear.
- Improve the efficiency of the parallel watershed algorithm to better perform the labeling of new pixels at a given threshold level.
- add a new algorithm to the parallel watershed to exploit contextual information about existing road candidates in order to straighten the extracted roads.

Appendix A

Parallel Algorithms Implementation

In this appendix, we concentrate on the implementation in Intelligent C of the main algorithms used in the linear feature extraction system developed for the purpose of detecting road networks in aerial images.

A.1 GeoFind Object Model

The Geofind system has been developed based on an object model illustrated in Figure 4.2 (top level only). The recognition system encompasses the following steps (see Figure 4.1:

pre-processing the original aerial image is inverted if necessary to ensure that the features of interest are lighter than the background. Brightness and contrast adjustment is performed. If more than one channel is available, addition and substraction between channels can be performed.

filtering the parallel DRO algorithm is used to create the marking function.

segmentation the parallel watershed algorithm is used to extract the linear features

post-processing contextual information is used to eliminate superfluous lines and to connect road segments.

The next sections give the Intelligent C implementation of the steps outlined above. Please refer to Appendix B for a complete description of the GeoFind system functionality.

A.2 Image Pre-Processing

The first step in the recognition process is the *pre-processing* step. In this step, the original image is filtered to accentuate the features of interest. The available operations are:

invert The parallel implementation of the *inversion* operation is shown below:

```
not8(Chan0(self),0,Result(self),0);
```

brightness/contrast The brightness w and contrast b parameters are used to transform the original image as follows:

$$y = 255/(w-b) \times x - 255 \times b/(w-b) \tag{A.1}$$

The parallel implementation of the *brightness* and *contrast* operations is given below:

```
for(i=0;i<256;i++) {
  tempVal = 255.0/(w-b)*i-255.0*b/(w-b);
  if (tempVal < 0.0) value=0;
  else if(tempVal > 254.0) value=254;
  else value = (int) tempVal;
  lut_value(self->tbl,i,value);
}
lut(self->tbl,source,0,dest,0);
```

add/subtract The parallel implementation of the *add* operation is shown below:

- Add or subtract a constant value *constantValue1* from the *source1* channel

```
if (self->operationOption1) {
  addkx(source1,0,self->constantValue1,
        source1,0,temp,0);
  for (i=0;i<8;i++)
    or(PLN(source1,i),0,temp,0,PLN(source1,i),0);
}
```

```
else {
  subkx(source1,0,self->constantValue1,
        source1,0,temp,0);
  not(temp,0,temp,0);
  for (i=0;i<8;i++)
    and(PLN(source1,i),0,temp,0,PLN(source1,i),0);
}
```

- Add the two channels *source1* and *source2*

```
addx(source1,0,source2,0,dest,0,temp,0);
for (i=0;i<8;i++)
  or(PLN(dest,i),0,temp,0,PLN(dest,i),0);
```

- Add or subtract a constant value *constantValue2* from the result image *dest*

```
if (self->operationOption2) {
  addkx(dest,0,self->constantValue2,
        dest,0,temp,0);
  for (i=0;i<8;i++)
    or(PLN(dest,i),0,temp,0,PLN(dest,i),0);
}
else {
  subkx(dest,0,self->constantValue2,
        dest,0,temp,0);
  not(temp,0,temp,0);
  for (i=0;i<8;i++)
    and(PLN(dest,i),0,temp,0,PLN(dest,i),0);
}
```

The parallel implementation of the *subtract* operation is shown below:

- Add or subtract a constant value *constantValue1* from the *source1* channel

```
if (self->operationOption1) {
  addkx(source1,0,self->constantValue1,
        source1,0,temp,0);
  for (i=0;i<8;i++)
    or(PLN(source1,i),0,temp,0,PLN(source1,i),0);
}
else {
  subkx(source1,0,self->constantValue1,
        source1,0,temp,0);
  not(temp,0,temp,0);
  for (i=0;i<8;i++)
    and(PLN(source1,i),0,temp,0,PLN(source1,i),0);
}
```

- Subtract the two channels *source1* and *source2*

```
subx(source1,0,source2,0,dest,0,temp,0);
not(temp,0,temp,0);
for (i=0;i<8;i++)
  and(PLN(dest,i),0,temp,0,PLN(dest,i),0);
```

- Add or subtract a constant value *constantValue2* from the result image *dest*

```
if (self->operationOption2) {
```

```
    addkx(dest,0,self->constantValue2,
          dest,0,temp,0);
    for (i=0;i<8;i++)
      or(PLN(dest,i),0,temp,0,PLN(dest,i),0);
  }
  else {
    subkx(dest,0,self->constantValue2,
          dest,0,temp,0);
    not(temp,0,temp,0);
    for (i=0;i<8;i++)
      and(PLN(dest,i),0,temp,0,PLN(dest,i),0);
  }
```

A.3 Image Filtering

The next step in the recognition process is the filtering operation. The marking function calculated in this step is used to identify the approximate location of the elongated features. The result of this filter operation will be used as the the initial condition for the watershed transformation to follow. The following filters are available for the calculation of the marking function:

- Parallel Duda Road Operator (DRO) filter
- Parabola filter
- Secant filter

◇ Sinc filter

◇ Gauss filter (one peak)

◇ Gauss filter (two peaks)

◇ Gauss filter (mathematical morphology)

◇ Average filter

◇ Laplace filter

◇ Marr-Hilldred filter

From Chapter 3, we recall that the the parallel DRO filter uses a scoring function $SCORE$, given in equation 3.2, for the detection of linear features. This function is calculated for each of the four masks (horizontal, vertical, right diagonal, and left diagonal), with the maximum value $SCORE_{max}$ replacing the intensity value of a given pixel a_2 in the center of the mask.

The parallel implementation of the parallel DRO filter (see Figure 3.3) is given below:

◇ create the look-up tables for *F(u)* and *G(u)*.

- Create the look-up table *tbl_F*

```
for(i=0;i<256;i++) {
  if (i < (int)(self->dudaTheta))
```

```
    cc = (int)(FACTOR*(M-((M_SIXTH)*
              (i/self->dudaTheta))));
  else cc = (int)(SIXTH*FACTOR);
  lut_value(self->tbl_F,i,cc);
}
```

- Create the look-up table *tbl_G*

```
for(i=0;i<256;i++) {
  if (i < (int)(self->dudaTheta1)) cc = FACTOR;
  else if(i <= (int)(self->dudaTheta2))
    cc = (int)(FACTOR*(1-ONE_E*(i-self->dudaTheta1)/
              (self->dudaTheta2 - self->dudaTheta1)));
  else if(i > (int)(self->dudaTheta2))
    cc = (int)(FACTOR*SIXTH);
  lut_value(self->tbl_G,i,cc);
}
```

◇ Calculate the value of the $SCORE_H$ function in the horizontal direction.

- Calculate $G(|\, a_1 - a_2 \,|)$

```
subx(Result(self),0,self->a0Frame,0,
     self->a1Frame,0,C(self),0);
sub(self->a0Frame,0,Result(self),0,
    self->tmp1Frame,0);
crep(self->a1Frame,0,self->tmp1Frame,0,
     C(self),0,self->tempFrame,0);
lut(self->tbl_G,self->tempFrame,0,self->a0Frame,0);
```

- Calculate $G(\mid a_2 - a_3 \mid)$

```
subx(Result(self),0,self->a1Frame,0,
     self->tmp1Frame,0,C(self),0);
sub(self->a1Frame,0,Result(self),0,
    self->tempFrame,0);
crep(self->tmp1Frame,0,self->tempFrame,0,
     C(self),0,self->multFrame,0);
lut(self->tbl_G,self->multFrame,0,self->a1Frame,0);
```

- Multiply the above two results $G(\mid a_1 - a_2 \mid) \times G(\mid a_2 - a_3 \mid)$

```
for (ii=0;ii<8;ii++) {
  for (jj=0;jj<8;jj++) {
    and(PLN(self->a0Frame,jj),0,PLN(self->a1Frame,ii),0,
        C(self),0);
    or(C(self),0,PLN(self->tmp1Frame,jj),0,
       PLN(self->tmp1Frame,jj),0);
  }
  for (kk=0;kk<ii;kk++)
    x2(self->tmp1Frame,0,self->tmp1Frame,0);

  add(self->multFrame,0,self->tmp1Frame,0,
      self->multFrame,0);
  zeros8(self->tmp1Frame,0);
}
```

- Calculate $F(a_i - b_i)$

```
tfr8(Result(self),0,self->cFrame,0);
```

```
subx(Result(self),0,self->cFrame,-2,
     self->tempFrame,0,C(self),0);
for (i=0;i<8;i++)
  not_and(C(self),0,PLN(self->tempFrame,i),0,
          PLN(self->tempFrame,i),0);

lut(self->tbl_F,self->tempFrame,0,self->bFrame,0);
```

- Calculate $\sum_{i=1}^{3} F(a_i - b_i)$

```
tfr8(f1,0,tmp1,tmp1EWshift);
addx(f1,0,tmp1,tmp1NSshift,f1,0,
     tmp2,0);
for (i=0;i<8;i++)
  or(tmp2,0,PLN(f1,i),0,PLN(f1,i),0);

tfr8(tmp1,0,tmp1,(tmp1EWshift*(-2)));
addx(f1,0,tmp1,(tmp1NSshift*(-2)),r,0,
     tmp2,0);
for (i=0;i<8;i++)
  or(tmp2,0,PLN(r,i),0,PLN(r,i),0);
```

- Similarly, calculate $\sum_{i=1}^{3} F(a_i - c_i)$, then add the two results

```
addx(self->tmp1Frame,0,self->cFrame,0,
     self->tempFrame,0,C(self),0);
for (i=0;i<8;i++)
  or(C(self),0,PLN(self->tempFrame,i),0,
     PLN(self->tempFrame,i),0);
```

- Finally, perform the division $G(|\ a_1 - a_2\ |) \times G(|\ a_2 - a_3\ |) / \sum_{i=1}^{3}(F(a_i - b_i) + F(a_i - c_i))\ |$. The division operation is implemented as a repeated subtract operation with carry.

  - First, subtract once and initialize the division counter to 1.

```
subx(self->multFrame,0,self->tempFrame,0,
     self->tmp1Frame,0,C(self),0);
for (j=0;j<8;j++)
  not_and(C(self),0,PLN(self->tmp1Frame,j),0,
          PLN(self->tmp1Frame,j),0);
```

  - Increase the *divnFrame* values by one

```
not(C(self),0,C(self),0);
MakeGuide(self->point,C(self),
          WindowFor(self->client),
          ScratchFrame2(self->client));

or(PLN(self->counterFrame,0),0,C(self),0,
       PLN(self->counterFrame,0),0);
add(self->divnFrame,0,self->counterFrame,0,
    self->divnFrame,0);
```

  - Continue to perform repeated subtract operations while increasing the division counter until idempotence.

```
while ((Extract1stGuide(self->point,C(self),
        WindowFor(self->client),
        ScratchFrame2(self->client)) > 0) &&
       (self->dudaMax < 254)) {
```

    Subtract once

```
subx(self->tmp1Frame,0,self->tempFrame,0,
     self->tmp1Frame,0,C(self),0);
for (j=0;j<8;j++)
  not_and(C(self),0,PLN(self->tmp1Frame,j),0,
          PLN(self->tmp1Frame,j),0);
```

Increase the divnFrame values by one.

```
zeros8(self->counterFrame,0);
not(C(self),0,C(self),0);
MakeGuide(self->point,C(self),
          WindowFor(self->client),
          ScratchFrame2(self->client));

or(PLN(self->counterFrame,0),0,C(self),0,
       PLN(self->counterFrame,0),0);
add(self->divnFrame,0,self->counterFrame,0,
    self->divnFrame,0);
```

Increment the counter.

```
(self->dudaMax)++;
}
```

◇ Once the $SCORE_H$ function has been calculated. we normalize the resulting image by using the formula:

$$y = 255/dudaMax \times i \tag{A.2}$$

```
for(i=0;i<256;i++) {
  if (i == 0) value=0;
```

```
    else if (i <= self->dudaMax)
      value = (int)((255.0/self->dudaMax)*i);
    else value = i;
    lut_value(self->tbl,i,value);
  }
  tfr8(self->divnFrame,0,self->tmp1Frame,0);
  lut(self->tbl,self->tmp1Frame,0,self->divnFrame,0);
```

◇ The last step involves thresholding the parallel DRO filter output.

- First, we enhance the parallel DRO filter output in anticipation of the thresholding operation by using the brightness w and contrast b parameters again. These parameters were used previously in the pre-processing stage.
- Second, we threshold the result image using an adaptive thresholding algorithm

```
Extract(self->dudaHistogram,
        self->divnFrame,
        WindowFor(self->client));

tdiv = 256/Capacity(self->dudaHistogram);
thr = QuickBimodalDivision(self->dudaHistogram,50);

if (self->dudaContrast <= 50)
  DudaThresholdIs(self,self->dudaContrast*
                  thr*tdiv/50);
```

```
else
  DudaThresholdIs(self,self->dudaContrast*
                  (255-thr*tdiv)/
                  50+255-2*(255-thr*tdiv));

gtk(self->divnFrame,0,DudaThreshold(self),
    PLN(Edges_4bitFrame(self),0),0);
```

Finally, we use contextual information to improve the results of the parallel DRO filter.

- Reduce thick lines.

```
if (c>1) {
  switch(dir) {
    case 0:
      r_n(f1,0,f1,0);
      r_s(f1,0,f1,0);break;
    case 1:
      r_e(f1,0,f1,0);
      r_w(f1,0,f1,0);break;
    case 2:
      r_n(f1,0,f1,0);
      r_w(f1,0,f1,0);break;
    case 3:
      r_n(f1,0,f1,0);
      r_e(f1,0,f1,0);break;
    default: break;
  }
```

```
}
```

- Eliminate dots.

  Eliminate dots in source (this includes diagonal lines) and save in *tmp*.

```
nodot(source,0,dest,0);
xor(source,0,dest,0,tmp1,0);
```

  Detect left diagonal lines and re-add to *dest* frame

```
d_n(tmp1,0,tmp2,0);
e_e(tmp2,0,tmp2,0);
d_all(tmp2,0,tmp2,0);
and(tmp1,0,tmp2,0,tmp2,0);
or(dest,0,tmp2,0,dest,0);
```

- Perform morphological closings in the *horizontal* direction.

```
for (i=1;i<=c;i++) {
  switch(dir) {
    case 0:
      if (i%a == 0) d_ns(f1,0,f1,0);
      d_ew(f1,0,f1,0);break;
    case 1:
      if (i%a == 0) d_ew(f1,0,f1,0);
      d_ns(f1,0,f1,0);break;
    case 2:
      if (i%a == 0) {
        tfr(f1,1,tmp,-1);
        or(f1,0,tmp,0,f1,0);
```

```
        tfr(f1,-1,tmp,1);
        or(f1,0,tmp,0,f1,0);
      }
      tfr(f1,1,tmp,1);
      or(f1,0,tmp,0,f1,0);
      tfr(f1,-1,tmp,-1);
      or(f1,0,tmp,0,f1,0);break;
    case 3:
      if (i%a == 0) {
        tfr(f1,1,tmp,1);
        or(f1,0,tmp,0,f1,0);
        tfr(f1,-1,tmp,-1);
        or(f1,0,tmp,0,f1,0);
      }
      tfr(f1,1,tmp,-1);
      or(f1,0,tmp,0,f1,0);
      tfr(f1,-1,tmp,1);
      or(f1,0,tmp,0,f1,0);break;
    default: break;
  }
}

for (i=0;i<c;i++) reduce(f1,0,f1,0);
```

- Eliminate small thin lines.

```
if (c>1) {
  reduce(f1,0,f1,0);
```

Eliminate dots in f1 (this includes diagonal lines) and save in *tmp*

```
nodot(f1,0,f1,0);
xor(f1,0,f1,0,tmp1,0);
```

Detect left diagonal lines and re-add to dest frame

```
d_n(tmp1,0,tmp2,0);
e_e(tmp2,0,tmp2,0);
d_all(tmp2,0,tmp2,0);
and(tmp1,0,tmp2,0,tmp2,0);
or(dest,0,tmp2,0,dest,0);

switch(dir) {
  case 0:
    d_ew(f1,0,f1,0);break;
  case 1:
    d_ns(f1,0,f1,0);break;
  case 2:
    tfr(f1,1,tmp,1);
    or(f1,0,tmp,0,f1,0);
    tfr(f1,-1,tmp,-1);
    or(f1,0,tmp,0,f1,0);break;
  case 3:
    tfr(f1,1,tmp,-1);
    or(f1,0,tmp,0,f1,0);
    tfr(f1,-1,tmp,1);
    or(f1,0,tmp,0,f1,0);break;
  default: break;
```

```
    }
  }
```

Store result in *Dest*.

```
tfr8(self->divnFrame,0,Dest(self),0);
```

◇ Similarly, the above procedure is applied in the other directions. At the end of this step, the grayscale of each pixel is replaced with $SCORE_{max}$ given by:

$$SCORE_{max}[A](x) = max\{SCORE \mid (x, SCORE) \in A\} \qquad (A.3)$$

A.4 Image Segmentation

The next step in the recognition process involves the extraction of the features of interest (roads) from the image. The algorithm developed for this task is the parallel watershed transformation. The starting point for the transformation is the filtered image from the previous step.

From Chapter 3, we recall that the parallel watershed algorithm is based on the definitions given in section 3.4.2. Therefore, one has to consider the successive thresholds of the image under study, and to compute geodesic influence zones of one threshold inside the next. The parallel algorithm is thus centered around the progressive flooding of the catchment basins of the image.

The parallel implementation of the parallel DRO filter (see Figure 3.11) is shown below:

◇ Increase the contrast of the input and create the label image.

- Create the 17 bit label image which is used to label new pixels at threshold level h. The depth of this label image is such that it allows unique labeling of pixels in a 512×256 image.

```
get_pmem_config(&wd,&ht,&plnDepth);

if (wd > ht) fastmask(f_16,(int)(wd/2),wd-5,5,ht-5);

k = 0;
l = 0;

for (i=5;i<ht-5;i++) {
  for (j=5;j<wd-5;j++) {
    pwait();
    setbyte(k,f_0to7,j,i);
    pwait();
    setbyte(l,f_8to15,j,i);
    l++;
    if (l>255) l = 0;
  }
  l = 0;
  k++;
  if (k>255) k = 0;
```

```
}
```

- Add the value δ to the grayscale value $im_i(p)$ of each pixel p in the input image im_i.

  Since the parallel watershed algorithm attempts to identify all the roads in the image — i.e. bright linear features, increasing the brightness of the image by the use of the linear filter:

$$im_i(p) = im_i(p) + \delta \tag{A.4}$$

  has the effect of strengthening the bright features in the grayscale image im_i before the watershed algorithm begins.

```
for(i=0;i<256;i++) {
  val=i+self->watershedDelta;
  if(val > 254) val=254;
  if(val < 0) val=0;
  lut_value(tbl,i,val);
}
lut(tbl,Result(self),0,Result(self),0);
```

◇ Suppose the flooding has been done up to a given level i. Every catchment basin already discovered — i.e., every catchment basin whose corresponding minimum has an altitude lower or equal to i — is supposed to have an unique label. The remaining of this section will show the implementation of the parallel watershed algorithm for one flooding level. The parallel watershed transformation repeats these steps $for i \leftarrow i_{min}$ to i_{max}, in our case $(0, 255)$.

- Add one flooding step *i-watershedStep* and get the next flooding step *i*.

  Identify new pixels only in frame *waTmp3*.

```
ltk(r,0,i-self->watershedStep,self->waTmp,0);
ltk(r,0,i,self->waTmp3,0);
```

  - Identify new pixels only in *waTmp3* enddescription

```
xor(self->waTmp,0,self->waTmp3,0,self->waTmp3,0);
```

  - Also eliminate the watershed points found in B from the GIZs found in *waTmp*.

```
not_and(B(self),0,self->waTmp,0,self->waTmp,0);
not_and(B(self),0,self->waTmp3,0,self->waTmp3,0);

MakeGuide(self->point,self->waTmp3,
          WindowFor(self),D(self));
```

  - Flag new points found in *waTmp3* in *watArea*

```
self->watArea = ExtractistGuide(self->point,
                                self->waTmp3,
                                WindowFor(self),D(self));
```

- While new points exist in image *watArea* DO:

  - Conditional dilate once the existing GIZ points found in *waTmp* using the new points found in *waTmp3* as the conditional frame. Separate these points in *waTmp2*.

    Dilate once *waTmp* into *waTmp2*.

```
Cd1_8dirIntoWith(self,self->waTmp,
                 self->waTmp2,self->waTmp3);
```

Separate new points obtained in the dilate and store in *waTmp2*.

```
xor(self->waTmp2,0,self->waTmp,0,self->waTmp2,0);
```

Complete the dilate operation on the GIZs found in *waTmp*.

```
or(self->waTmp,0,self->waTmp2,0,self->waTmp,0);
```

Extract first point.

```
MakeGuide(self->point,self->waTmp2,
          WindowFor(self),D(self));
```

- The points that are adjacent to existing GIZs are included in their respective GIZs (i.e. they will be labeled respectively) in frame *waTmp* using the 17 bit label image.

  Eliminate watershed points from *waTmp2*.

  ```
  not_and(B(self),0,self->waTmp2,0,self->waTmp2,0);
  ```

  Label these points with their respective GIZs labels using the 17 bit label image.

  ```
  for (j=0;j<8;j++) {
    Cd1_8dirIntoWith(self,
        PLN(self->waLabel0to7,j),
        PLN(self->waLabel0to7,j),
        self->waTmp2);
    Cd1_8dirIntoWith(self,
        PLN(self->waLabel8to15,j),
        PLN(self->waLabel8to15,j),
        self->waTmp2);
  }
  Cd1_8dirIntoWith(self,self->waLabel16,
  ```

```
self->waLabel16,self->waTmp2);
```

- Identify the watershed points amongst these dilate points using the fact that each new dilate point has a label which has the following properties:

  1. if the point is adjacent to only one GIZ then it's new label corresponds to the label of the GIZ (i.e. GIZ point).
  2. if the point is adjacent to more than one GIZ then it's new label corresponds to the label of one of the GIZs or the label is an entirely different label (i.e. watershed point).

  To identify if a point is a watershed point we must exploit the above properties. Thus we compare the label of each point to its eight close neighbours and identify if this point has a label which is similar to any adjacent GIZ label. If a difference occurs we mark this point as a watershed point.

  This is the heart of the parallel watershed transformation.

```
test2 = 1;
while (test2 && oldtest) {
  tfr(self->waTmp2,0,D(self),0);
```

Reduce D to idempotence.

```
  j = 1;
  while (j) {
```

Identify all the new seeds for the new GIZs. Repeating the reduce operation five times diminishes the time required for idempotence.

```
    reduce(D(self),0,C(self),0);
```

```
    reduce(C(self),0,C(self),0);
    reduce(C(self),0,C(self),0);
    reduce(C(self),0,C(self),0);
    reduce(C(self),0,C(self),0);
    xor(D(self),0,C(self),0,F(self),0);
    tfr(C(self),0,D(self),0);
    MakeGuide(self->point,F(self),
              WindowFor(self),C(self));

    j = Extract1stGuide(self->point,F(self),
        WindowFor(self),C(self));
  }
```

Identify the labels of the new points in D.

```
  for (i=0;i<8;i++) {
    and(PLN(self->waLabel0to7,i),0,D(self),0,
        PLN(self->waTmpLabel0to7,i),0);
    and(PLN(self->waLabel8to15,i),0,D(self),0,
        PLN(self->waTmpLabel8to15,i),0);
  }
  and(self->waLabel16,0,D(self),0,
      self->waTmpLabel16,0);
```

- Compare the label of point P with the labels of its eight neighbours. If a label of any of the eight neighbours is different, flag the difference. In this case, point P is a watershed point.

  The implementation is shown below for one of the eight neighbors only (neighbor 0). The same algorithm applies to all the eight close

neighbors.

Shift P to position *0*.

```
tfr8(self->waTmpLabel0to7, 1,
     self->waTmpLabel0to7, -1);
tfr8(self->waTmpLabel8to15,1,
     self->waTmpLabel8to15,-1);
tfr(self->waTmpLabel16,    1,
    self->waTmpLabel16,   -1);
```

Shift D.

```
tfr(D(self),1,D(self),-1);
```

Compare P with label at position *0*.

```
for (i=0;i<8;i++) {
  xor(PLN(self->waLabel0to7,i),0,
      PLN(self->waTmpLabel0to7,i),0,
      tmp,0);
  or(tmp,0,tmp2,0,tmp2,0);
  xor(PLN(self->waLabel8to15,i),0,
      PLN(self->waTmpLabel8to15,i),0,
      tmp,0);
  or(tmp,0,tmp2,0,tmp2,0);
}
xor(self->waLabel16,0,
    self->waTmpLabel16,0,tmp,0);
or(tmp,0,tmp2,0,tmp2,0);
```

Mask out the neighbours with label *0*.

```
and(self->waTmp,0,tmp2,0,tmp2,0);
```

Mask out neighbours which are watershed points.

```
not_and(B(self),0,tmp2,0,tmp2,0);
```

Mask out the result using the shifted D.

```
and(tmp2,0,D(self),0,tmp2,0);
```

Shift result of compare operation back to position P and store in frame *watResult*.

```
tfr(tmp2,-1,tmp2,1);
or(tmp2,0,watResult,0,watResult,0);
```

- At the end of the above step, points P have been compared with all their eight close neighbors. The flagged points amongst these are watershed points.

  Add these points to the watershed frame B.

  ```
  or(B(self),0,watResult,0,B(self),0);
  ```

  Eliminate these points from the GIZs in *waTmp*. This operation is necessary to eliminate the following scenario:

  — a new watershed candidate is labeled by a conditional dilate of the GIZ label conditional on the *waTmp* footprint. If watershed points are allowed to survive as parts of the GIZs then the labels of these watershed points can corrupt the labeling operation outlined above.

  ```
  xor(self->waTmp,0,watResult,0,self->waTmp,0);
  ```

  Eliminate the labels of these points from the *label* frame

  ```
  for (i=0;i<8;i++) {
  ```

```
    not_and(watResult,0,PLN(self->waLabel0to7,i),0,
            PLN(self->waLabel0to7,i),0);
    not_and(watResult,0,PLN(self->waLabel8to15,i),0,
            PLN(self->waLabel8to15,i),0);
  }
  not_and(watResult,0,self->waLabel16,0,
          self->waLabel16,0);
```

- Remove these new found watershed points from *waTmp2* and *waTmp3*.

```
  xor(self->waTmp2,0,D(self),0,self->waTmp2,0);

  xor(self->waTmp3,0,D(self),0,self->waTmp3,0);

  MakeGuide(self->point,self->waTmp2,
            WindowFor(self),F(self));
```

Any points left in *waTmp2* ?

```
  test2 = Extract1stGuide(self->point,
                          self->waTmp2,
                          WindowFor(self),
                          F(self));
}
```

The *waTmp2* points have been removed from the new points in *waTmp3*.

```
MakeGuide(self->point,self->waTmp3,
          WindowFor(self),D(self));
```

Any new points in *waTmp3* ?

```
test = Extract1stGuide(self->point,
```

```
                                self->waTmp3,
                                WindowFor(self),
                                D(self));
    self->watArea = test;
```

- Examine if any points have been left in *waTmp3*

  More points (i.e. seeds) left in *waTmp3* ? If so, take each point (i.e. seeds), add it to *waTmp*, label it, and label its neighbours. Identify all the new seeds.

```
  tfr(self->waTmp3,0,D(self),0);
  j = 1;
  while (j) {
```

  Reduce D 5 times. Check for idempotence in F.

```
    reduce(D(self),0,C(self),0);
    reduce(C(self),0,C(self),0);
    reduce(C(self),0,C(self),0);
    reduce(C(self),0,C(self),0);
    reduce(C(self),0,C(self),0);
    xor(D(self),0,C(self),0,F(self),0);
    tfr(C(self),0,D(self),0);
    MakeGuide(self->point,F(self),
              WindowFor(self),C(self));
```

  Check for idempotence.

```
    j = Extract1stGuide(self->point,F(self),
                        WindowFor(self),C(self));
  }
```

  Test for presence of closed contours in new GIZs:

1. remove dots from D and place result in C
2. D now contains only seed points while C contains all the closed polygons
3. make guide on C to show any closed contours

```
nodot(D(self),0,C(self),0);
MakeGuide(self->point,C(self),
          WindowFor(self),F(self));
```

4. check for the presence of closed contours in C
5. if closed contours exist, replace them with unique seeds

While close contours exist in frame C.

```
while (Extract1stGuide(self->point,C(self),
       WindowFor(self),F(self))) {
```

Separate this point in new frame G.

```
  zeros(G(self),0);
  pwait();
  setbit(1,G(self),PT_X(self->point),
         PT_Y(self->point));
```

Add this new seed to D.

```
  or(D(self),0,G(self),0,D(self),0);
```

Conditional dilate this point in G with the closed polygon in C until idempotence.

```
  j = 1;
  while (j) {
```

Conditional dilate G with C to separate the respective closed polygon

in C. Repeating the conditional dilate operation five times diminishes the time required for idempotence.

```
Cd1_8dirIntoWith(self,G(self),
                    F(self),C(self));
```

dilate once in F

```
for (i=0;i<4;i++)
  Cd1_8dirIntoWith(self,F(self),
                      F(self),C(self));
```

dilate in F

```
xor(G(self),0,F(self),0,H(self),0);
tfr(F(self),0,G(self),0);
zeros(F(self),0);
MakeGuide(self->point,H(self),
          WindowFor(self),F(self));
```

check for idempotence.

```
j = Extract1stGuide(self->point,
                    H(self),
                    WindowFor(self),
                    F(self));
```

xor the above extracted polygon from C.

```
xor(C(self),0,F(self),0,C(self));
```

Any more closed polygons in C ?

```
MakeGuide(self->point,C(self),
          WindowFor(self),F(self));
}
```

Label these new seeds.

```
for (i=0;i<8;i++) {
  and(PLN(self->label0to7_8bitFrame,i),0,
      D(self),0,
      PLN(self->waTmpLabel0to7,i),0);
  and(PLN(self->label8to15_8bitFrame,i),0,
      D(self),0,
      PLN(self->waTmpLabel8to15,i),0);
}
and(self->label16_1bitFrame,0,D(self),0,
    self->waTmpLabel16,0);
```

Label the neighbours of the new seeds.

```
j = 1;
while (j) {
```

Label the neighbours of the new seeds also by conditional dilate of the label frame twice.

By dilating five times before doing the idempotency check. time is saved.

```
    for (i=0;i<8;i++) {
      Cd1_8dirIntoWith(self,
          PLN(self->waTmpLabel0to7,i),
          PLN(self->waTmpLabel0to7,i),
          self->waTmp3);
      Cd1_8dirIntoWith(self,
          PLN(self->waTmpLabel0to7,i),
          PLN(self->waTmpLabel0to7,i),
```

```
    self->waTmp3);
Cd1_8dirIntoWith(self,
    PLN(self->waTmpLabel0to7,i),
    PLN(self->waTmpLabel0to7,i),
    self->waTmp3);
Cd1_8dirIntoWith(self,
    PLN(self->waTmpLabel0to7,i),
    PLN(self->waTmpLabel0to7,i),
    self->waTmp3);
Cd1_8dirIntoWith(self,
    PLN(self->waTmpLabel0to7,i),
    PLN(self->waTmpLabel0to7,i),
    self->waTmp3);

Cd1_8dirIntoWith(self,
    PLN(self->waTmpLabel8to15,i),
    PLN(self->waTmpLabel8to15,i),
    self->waTmp3);
Cd1_8dirIntoWith(self,
    PLN(self->waTmpLabel8to15,i),
    PLN(self->waTmpLabel8to15,i),
    self->waTmp3);
Cd1_8dirIntoWith(self,
    PLN(self->waTmpLabel8to15,i),
    PLN(self->waTmpLabel8to15,i),
    self->waTmp3);
Cd1_8dirIntoWith(self,
```

```
        PLN(self->waTmpLabel8to15,i),
        PLN(self->waTmpLabel8to15,i),
        self->waTmp3);
    Cd1_8dirIntoWith(self,
        PLN(self->waTmpLabel8to15,i),
        PLN(self->waTmpLabel8to15,i),
        self->waTmp3);
}
Cd1_8dirIntoWith(self,self->waTmpLabel16,
        self->waTmpLabel16,self->waTmp3);
Cd1_8dirIntoWith(self,self->waTmpLabel16,
        self->waTmpLabel16,self->waTmp3);
Cd1_8dirIntoWith(self,self->waTmpLabel16,
        self->waTmpLabel16,self->waTmp3);
Cd1_8dirIntoWith(self,self->waTmpLabel16,
        self->waTmpLabel16,self->waTmp3);
Cd1_8dirIntoWith(self,self->waTmpLabel16,
        self->waTmpLabel16,self->waTmp3);

Cd1_8dirIntoWith(self,D(self),
        C(self),self->waTmp3);
Cd1_8dirIntoWith(self,C(self),
        C(self),self->waTmp3);
Cd1_8dirIntoWith(self,C(self),
        C(self),self->waTmp3);
Cd1_8dirIntoWith(self,C(self),
        C(self),self->waTmp3);
```

```
      Cd1_8dirIntoWith(self,C(self),
            C(self),self->waTmp3);

      xor(D(self),0,C(self),0,F(self),0);
      tfr(C(self),0,D(self),0);
      MakeGuide(self->point,F(self),
            WindowFor(self),C(self));

      j = Extract1stGuide(self->point,F(self),
            WindowFor(self),C(self));
    }
```

Add these new GIZs to *waTmp*.

```
    for (i=0;i<8;i++) {
      or(PLN(self->waTmpLabel0to7,i),0,
         PLN(self->waLabel0to7,i),0,
         PLN(self->waLabel0to7,i),0);
      or(PLN(self->waTmpLabel8to15,i),0,
         PLN(self->waLabel8to15,i),0,
         PLN(self->waLabel8to15,i),0);
    }
    or(self->waTmpLabel16,0,self->waLabel16,0,
       self->waLabel16,0);

    self->watArea = 0;
  }
}
```

A.5 Image Post-processing

In this, the last step in the road detection system, contextual information is being used to connect colinear road segments and to eliminate unwanted small linear features.

The parallel implementation (see Figure 3.11) is given below:

◇ First eliminate small closed contours

```
tfr(f,0,temp,0);
for (i=0;i<self->watershedGrid;i++)
  reduce(temp,0,temp,0);
nodot(temp,0,temp,0);
for (i=0;i<self->watershedGrid;i++)
  Cd1_8dirIntoWith(self,temp,temp,f);
not(temp,0,temp,0);
not(f,0,f,0);
for (i=0;i<self->watershedGrid+2;i++)
  reduce(temp,0,temp,0);
nodot(temp,0,temp,0);
tfr(temp,0,f,0);
```

◇ Next, eliminate small watershed segments

```
for (i=0;i<self->watershedGrid;i++)
  reduce(temp,0,temp,0);
```

```
nodot(temp,0,temp,0);
for (i=0;i<self->watershedGrid;i++)
  Cd1_8dirIntoWith(self,temp,temp,f);
```

◇ Finally, trim existing roads

```
for (i=0;i<self->watershedGrid;i++)
  reduce(temp,0,temp,0);
```

Appendix B

GeoFind Functionality Examples

This Appendix presents several image processing applications which have been developed on the *GeoFind 2.4* system which runs on an AIS-3500 vision computer. These include:

aquisition Capturing images through a RS-170 CCD camera.

uploading/downloading Transfering images between the AIS-3500 and a SUN UNIX workstation.

video grab Viewing video images from a live camera and grabbing a video snapshot.

pre-processing Inverting an image, adjusting the contrast, simple thresholding, adding and subtracting channels.

filtering Applying different image filters to an image.

segmentation Adaptive thresholding, edge detection, watershed transformation, Hough transformation.

B.1 Image Acquisition

The *GeoFind 2.4* system displays 512×256 images. To capture graylevel video images from a RS-170 CCD camera, do the following:

- Select the camera channel by plugging the CCD camera into one of the camera inputs *(0 through 5)*.
- Click on the corresponding channel on the *GeoFind 2.4* banner *(a through f)*.
- Deselect the memory button on the *GeoFind 2.4* banner (i.e. ensure that the memory button is NOT highlighted in yellow).
- Highlight the *GeoFind 2.4* banner (i.e. ensure that the *GeoFind 2.4* banner IS highlighted in yellow).

The real-time video image should now appear on the screen. Use the camera adjustments to obtain a good image.

B.2 Image Uploading and Downloading

The AIS-3500 vision computer processes images in SUN raster format.

B.2.1 Downloading an Image

Downloading Using the umodem command

To *download* an image called *image.ras* from the host computer (SUN UNIX workstation) to the AIS-3500 vision computer:

- Ensure that there is enough space in the AIS-3500 memory to accomodate the image. The *GeoFind 2.4* application uses 341K leaving approximately 60K available for data files. A 200 × 300 image in SUN raster format occupies 60.8K of memory.

```
ais>ls
  rwx  1 340772/340772 - GeoFind
ais>
```

- Start *GeoFind 2.4* on the AIS-3500 vision computer.

```
ais> GeoFind
   Initializing GeoFind ver. 2.4
   Time:   46560.0 ms for CreateLabelFrame
number of blocks   = 0
capacity of block table = 50
base of bpc buffer = 00209500
```

```
top  of bpc buffer = 0023ffff
next free location = 00209500
bytes available    = 224000

   ***Reading font file gfinit.dat
```

- In a SUN *xterm* window, run the following command at the sun prompt:

```
sun> aterm <unix2ais
Copyright (c) 1988-1993, Applied Intelligent Systems, Inc.
All rights reserved
VERSION: 2.03
 No response from machine
 Check connections/press "ABORT"

cmd>!umodem -sbm result.ras </dev/ttya >/dev/ttya

UMODEM:  File Name: result.ras
Estimated File Size 1 Records, 0 Bytes

UMODEM:  8-Bit Transmission Enabled
UMODEM:  Ready to SEND File
```

where, *aterm* is a terminal emulation program available from AISI with the LAYERS development environment, and *unix2ais* is a script which invokes the **umodem** command:

```
^A!umodem -sbm result.ras </dev/ttya >/dev/ttya
```

- In *GeoFind 2.4* select *image* from the *File* pull down menu. and click on the $UNIXhost > AIS3500image$ button. When the image has been successfully downloaded the terminal program exists with `status = 0`.

The image has now been automatically loaded in the *a* channel. Note that the image name has to be *image.ras* in the *Image File* panel.

To view the image, select channel *a* and highlight the memory button on the *GeoFind 2.4* banner.

Downloading Using the `dlb` command

To *download* an image called *image.ras* from the host computer (SUN UNIX workstation) to the AIS-3500 vision computer:

- Ensure that there is enough space in the AIS-3500 memory to accomodate the image. The *GeoFind 2.4* application uses 341K leaving approximately 60K available for data files. A 200 × 300 image in SUN raster format occupies 60.8K of memory.

```
ais>ls
  rwx  1 340772/340772 - GeoFind
ais>
```

- The *GeoFind 2.4* on the AIS-3500 vision computer should not be running.

- In a SUN *xterm* window, run the following command at the sun prompt:

```
sun> aterm
Copyright (c) 1988-1993, Applied Intelligent Systems, Inc.
All rights reserved
VERSION: 2.03
ais> ^A
cmd>dlb <image file name> image.ras
%dlb image.ras
 Size = 60800 bytes.
  60800  -  Download time remaining =  0 Min.  0 Sec.
ais>
```

where, *aterm* is a terminal emulation program available from AISI with the LAYERS development environment.

- The image file will appear in the

usr directory on the AIS-3500 vision computer.

```
ais>ls
  rwx  1 340772/340772 - GeoFind
  rw-* 1 60800/60800 - image.ras
ais>
```

- Now, start *GeoFind 2.4*.

```
ais>GeoFind
   Initializing GeoFind ver. 2.4
```

```
   Time:   46560.0 ms for CreateLabelFrame
number of blocks   = 0
capacity of block table = 50
base of bpc buffer = 00209500
top  of bpc buffer = 0023ffff
next free location = 00209500
bytes available    = 224000

   ***Reading font file gfinit.dat
```

The image has now been automatically loaded in the *a* channel. Note that the image name has to be *image.ras* in the *Image File* panel.

To view the image, select channel *a* and highlight the memory button on the *GeoFind 2.4* banner.

B.2.2 Uploading an Image

To *upload* an image called *image.ras* from the AIS-3500 vision computer to the host computer (SUN UNIX workstation):

- Start *GeoFind 2.4* on the AIS-3500 vision computer.

```
ais> GeoFind
   Initializing GeoFind ver. 2.4
   Time:   46560.0 ms for CreateLabelFrame
```

```
number of blocks    = 0
capacity of block table = 50
base of bpc buffer = 00209500
top  of bpc buffer = 0023ffff
next free location = 00209500
bytes available     = 224000

   ***Reading font file gfinit.dat
```

- In a SUN *xterm* window, run the following command at the sun prompt:

```
sun> aterm <ais2unix
Copyright (c) 1988-1993, Applied Intelligent Systems, Inc.
All rights reserved
VERSION: 2.03
 No response from machine
 Check connections/press "ABORT"

cmd>!umodem -rbm result.ras </dev/ttya >/dev/ttya

UMODEM:  File Name: result.ras
UMODEM:  8-Bit Transmission Enabled
UMODEM:  Ready to RECEIVE File
```

where, *aterm* is a terminal emulation program available from AISI with the LAYERS development environment, and *ais2unix* is a script which invokes the umodem command:

```
~A!umodem -rbm result.ras </dev/ttya >/dev/ttya
```

- In *GeoFind 2.4*, select the image to be uploaded by displaying it on the screen and selecting the window to be downloaded by using the blue rectangle. Then, select the *image* panel button from the *File* pull-down menu, and click on the $AIS3500image > UNIXhost$ button. When the image has been successfully uploaded the terminal program exists with `status = 0`.

The image has now been automatically saved on the SUN workstation. Note that the image name has to be *image.ras* in the *Image File* panel.

B.3 Video Grab

To capture (i.e. upload an image called *image.ras* from the AIS-3500 vision computer to the host computer) a snapshot from a live image received from a CCD camera do the following:

- Start *GeoFind 2.4* on the AIS-3500 vision computer.

```
ais> GeoFind
   Initializing GeoFind ver. 2.4
   Time:   46560.0 ms for CreateLabelFrame
number of blocks   = 0
capacity of block table = 50
base of bpc buffer = 00209500
```

```
top  of bpc buffer = 0023ffff
next free location = 00209500
bytes available    = 224000

   ***Reading font file gfinit.dat
```

- In a SUN *xterm* window, run the following command at the sun prompt:

```
sun> aterm <ais2unix
Copyright (c) 1988-1993, Applied Intelligent Systems, Inc.
All rights reserved
VERSION: 2.03
 No response from machine
 Check connections/press "ABORT"

cmd>!umodem -rbm result.ras </dev/ttya >/dev/ttya

UMODEM:  File Name: result.ras
UMODEM:  8-Bit Transmission Enabled
UMODEM:  Ready to RECEIVE File
```

where, *aterm* is a terminal emulation program available from AISI with the LAYERS development environment, and *ais2unix* is a script which invokes the umodem command:

```
^A!umodem -rbm result.ras </dev/ttya >/dev/ttya
```

- In *GeoFind 2.4*, select the image to be uploaded by displaying it on the screen. The whole screen will be downloaded (i.e. 512 × 256. Then, select the *video* panel button from the *File* pull-down menu, and click on the $AIS3500image > UNIXhost$ button. When the image has been successfully uploaded the terminal program exists with `status = 0`.

The image has now been automatically saved on the SUN workstation. The size of the image is 512 × 256. Note that the image name has to be *image.ras* in the *Image File* panel.

B.4 Image Pre-processing

The following operations are available in the pre-processing stage:

invert inversion of an image.

threshold threshold the image.

brightness/contrast adjust the image quality.

add/subtract combine image channels.

B.4.1 Inverting an Image

To invert an image, select the *map* panel button from the *image* pull-down menu, and click on the *invert* button.

B.4.2 Thresholding an Image

The *Map* panel also includes a *threshold* feature. To threshold an image, first select the threshold value *(0, 255)* and second, select the threshold operation by clicking on the word *threshold*.

B.4.3 Adjusting an Image Quality

The following operations are available to adjust the quality of an image:

brightness this operation changes the value of h_{max} from 255 to a value between 0 and 255, in effect increasing the brightness of the image. To increase the brightness of the image, select the value, then click on the word *brightness*.

contrast this operation changes the value of h_{min} from 0 to a value between 0 and 255, in effect increasing the contrast of the image. To increase the contrast of the image, select the value, then click on the word *contrast*.

B.4.4 Combining Image Channels

Image channels can be added through the *Calculate* panel. The following operations are available between two channels:

addition

subtraction

B.5 Image Filtering

Different well known filtering algorithms can be applied to an image through the *Filter* panel. The list of available filters includes:

parallel Duda Road Operator

parabola

secant

sinc

Gauss

average

Laplace

Marr-Hildred

Sobel

B.5.1 Image Histogram

A parallel bimodal histogram algorithm is available to display the histogram of a given image through the *Histogram* panel.

B.6 Image Segmentation

Image segmentation using parallel adaptive thresholding is available in the *Segment* panel.

B.6.1 Adaptive Thresholding

A parallel adaptive thresholding algorithm is available in the *Adaptive Threshold* panel.

B.6.2 Edge Detection

A four directional parallel edge detection algorithm is available through the *Edge* panel.

B.6.3 Watershed Transformation

The parallel watershed transformation is performed through the *Watershed* panel. The transformation can be performed in *step* mode (one threshold level at a time), or in continuous mode. Different marking function levels are available.

B.6.4 Hough Transform

A parallel Hough transformation is available through the *Hough* panel.

B.7 GeoFind Debug Mode

To access different display levels during the execution of the different image processing applications, enter the *Debug* panel under the *window* pull-down menu and toggle the corresponding flags (i.e. hex digit). Note that when a flag is ON it is replaced by a *"!"* character. By default, all flags are turned OFF. Remember that displaying takes valuable CPU time, hence slowing down algorithm execution.

B.7.1 Debug Flags

The following section presents a list of the debug flags which can be used to turn on debug information.

Parallel DRO Filter

The following debug information can be invoked during the execution of the parallel DRO filter execution by using the *flag2* flags in the *Debug panel.*

0 Not used.

1 Not used.

2 Time delay to slow down the algorithm execution. Use the *delay* parameter in the *Debug* panel to set the delay time in seconds.

3 Print timing information at the serial port.

4 Show normalization to 255.

5 Show intermediate result frames.

6 Not used.

7 Execute the `DudaParAbs`*x*`Code` methods directly instead of the parallel blocks to show additional debugging information.

8 Execute the `DudaParMultCode` method directly instead of the parallel block to show additional debugging information.

9 Execute the `DudaParSubCode` and `DudaParSubInitCode` methods directly instead of the parallel blocks to show additional debugging information.

A Execute the DudaParF_*x*Code methods directly instead of the parallel blocks to show additional debugging information.

B Execute the DudaParResult*x*Code methods directly instead of the parallel blocks to show additional debugging information.

Parallel Watershed Transformation

The following debug information can be invoked during the execution of the parallel watershed transformation by using the *flag1* flags in the *Debug panel.*

0 Show watershed points in *red.*

1 Show previous geodesic influence zones (GIZ) in *cyan.*

2 Show new points at level h in *yellow.*

3 Print debug information at the serial port.

4 Print timing information at the serial port.

5 Show intermediate result frames.

Bibliography

[1] M. Abrams, A. Blusson, V. Carrere, T. Nguyen, and Y. Rabu. Image processing applications for geologic mapping. *IBM Journal of Research and Development*, 29(2):177–187, 1985.

[2] Applied Intelligent Systems Inc., 110 Parkland Plaza, Ann Arbor, MI 48103. *C Layer Reference Manual*. 2.03 edition, February 1993.

[3] Applied Intelligent Systems Inc., 110 Parkland Plaza, Ann Arbor, MI 48103. *Layers Installation and Users Guide*. 2.03 edition, February 1993.

[4] Applied Intelligent Systems Inc., 110 Parkland Plaza, Ann Arbor, MI 48103. *Object Layer Reference Manual*. 2.03 edition, February 1993.

[5] G.P. Ashkar and J.W. Modestino. The contour extraction problem with biomedical applications. *Computer Graphics and Image Processing*, 7:331–355, 1987.

[6] B. Bajcsy and M. Tavakoli. Computer recognition of roads from satellite pictures. *IEEE Transactions of Systems, Man, and Cybernetics*, SMC-6:623–637, 1976.

[7] M. Barzohar and D.B. Cooper. Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition - 1993*, pages 459–64, Los Alamitos, CA, USA, June 1993. IEEE Computer Society Press.

[8] S.P. Benjamin and L. Gaydos. Processing of scanned imagery for cartographic feature extraction. In R.M. Haralick, editor, *IEEE Pecora IX Symposium: Spatial Information Technologies for Remote Sensing Today and Tomorrow*, pages 222–230. IEEE Computer Society Press, October 1984.

[9] S. Beucher. Watersheds of functions and picture segmentation. In *Proceeedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1928–1931, Paris, France, May 1982.

[10] S. Beucher and C. Lantuéjoul. Use of watersheds in contour detection. In *Proceedings of the International Workshop on Image Processing, Real-Time Edge and Motion Detection/Estimation*, Rennes, France, Sept. 1979.

[11] J.B. Burns, A.R. Hanson, and E.M. Riseman. Extracting straight lines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(4):425–455, 1986.

[12] J.F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, PAMI-8:679–698, 1986.

[13] C.B. Chittineni. Edge and line detection on multidimensional noisy imagery data. *IEEE Transactions on Geoscience and Remote Sensing*, GE-21:163–174, 1983.

[14] B.J. Cox. *Object Oriented Programming - An Evolutionary Approach*. Addison Wesley, 1986.

[15] I. Destival. Mathematical morphology applied to remote sensing. *Acta Astronautica*, 13:371–385, 1986.

[16] H. Digabel and C. Lantuéjoul. Iterative algorithms. In J. L. Chermant, editor, *Proceedings of the 2nd European Symposium on Quantitative Analysis of Microstructures in Material Science, Biology and Medicine*, volume 1, pages 85–99, Stuttgart, Germany, Oct. 1978. Riederer Verlag.

[17] R.O. Duda and P.E. Hart. Use of hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[18] R. Ferrand and H. Marty. Computer processing test on a simulated spot image of Toulouse (France). *Photo-Interpretation*, 3:39–45, 1985.

[19] A.L. Fisher and P.T. Highnam. Computing the hough transform on a scan line array processor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(3):262–265, 1989.

[20] M. A. Fishler, J. M. Tenenbaum, and H. C. Wolf. Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge

integration technique. *Computer Graphics and Image Processing*, 15:201–223, 1981.

[21] F. Friedlander. A sequential algorithm for detecting watersheds on a gray level image. *Acta Stereologica (Proceedings of the 7th International Congress for Stereology)*, III(6):663–668, 1987.

[22] J. Gilli. Contribution of remote sensing data and geographic information systems to preliminary road layouts studies. In *Proceedings of the International Symposium on Remote Sensing of the Environment*, volume 1, pages 529–538, 1990.

[23] A.F.H. Goetz, J.B. Wellman, and William L. Barnes. Optical remote sensing of the earth. *Proc. IEEE*, 73(6):950–969, 1985.

[24] D.G. Goodenough, M. Goldberg, G. Plunkett, and J. Zelek. An expert system for remote sensing. *IEEE Transactions on Geoscience and Remote Sensing*, GE-25(3):349–359, 1987.

[25] W.D. Groch. Extraction of line shaped objects from aerial images using a special operator to analyze the profiles of functions. *Computer Graphics and Image Processing*, 18:347–358, 1982.

[26] B. Guindon. Multi-temporal scene analysis: A tool to aid in the identification of cartographically significant edge features on satellite imagery. *Canadian Journal of Remote Sensing*, 14:38–45, 1988.

[27] C.M. Gurney. Threshold selection for line detection algorithms. *IEEE Transactions on Geoscience and Remote Sensing*, GE-18:204–211, 1980.

[28] C.M. Gurney. The use of linear feature detection to investigate tm data performance and processing. In J. Barker, editor, *Proceedings of the LANDSAT-4 Science Characterization Early Results Symposium*, volume III, pages 513–526, NASA Goddard Space Flight Centre, MD, USA, December 1983. National Aeronautics and Space Administration.

[29] C.M. Gurney and R.G. Townshend. The use of contextual information in the classification of remotely sensed data. *Photogrammetric Engineering and Remote Sensing*, 49:55–64, 1983.

[30] J.R. Hall and F.C. Mertz. Edge discrimination as applied to thematic mapper data. In *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, volume 1, pages 7.1–7.4, New York, NY, USA, 1983. IEEE.

[31] J. Hu, B. Sakoda, and T. Pavlidis. Interactive road finding for aerial images. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, pages 56–63, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press.

[32] K. Hwang and F.A. Briggs. *Computer Architecture and Parallel Processing*. Computer Organization and Architecture. McGraw-Hill, 1984.

[33] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proceedings of the First International Conference on Computer Vision*, page 259, 1987.

[34] L. Lalitha. Technique for road detection from high resolution satellite images. In *IGARSS'89 - International Geoscience and Remote Sensing Symposium*, volume 4, pages 2246–2249, 1989.

[35] C. Lipari, M. Trivedi, and C. Harlow. Geometric modeling and recognition of elongated regions in aerial images. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1600–1612, 1989.

[36] R. Nevatia and K.R. Babu. Linear feature extraction and description. *Computer Graphics and Image Processing*, 13:257–269, 1980.

[37] G. Nicchiotti and E. Ottaviani. Automatic cartography: A 'snake' approach. In V. Cappellini and A.G. Constantinides, editors, *Proceedings of the International Conference on Digital Signal Processing - 91*, pages 591–594, Amsterdam, Netherlands, September 1991. Elsevier.

[38] B. Nicolin and R. Gabler. A knowledge-based system for the analysis of aerial images. *IEEE Transactions on Geoscience and Remote Sensing*, GE-25(3):317–329, 1987.

[39] D. O'Brien. Automatic extraction of road networks from digital imagery. In *Proceedings of the International Symposium on Topographic Applications of SPOT Data*, pages 273–287. Canadian Institute of Surveying and Mapping, October 1988.

[40] J.M. Ogier, C. Olivier, and Y. Lecourtier. Extraction of road networks from digitized maps. In J. Vandewalle, R. Boite, M. Moonen, and A. Ooster-

linck, editors. *Signal Processing VI - Theories and Applications. Proceedings of EUSIPCO-92, Sixth European Signal Processing Conference*, volume 1, pages 619–622, Amsterdam, Netherlands, August 1992. Elsevier.

[41] P.E. Pelletier. Spatial variation analysis of thematic mapper data for the identification of linear features in agriculture and landscapes. In R.M. Haralick, editor, *IEEE Pecora IX Symposium: Spatial Information Technologies for Remote Sensing Today and Tomorrow*, pages 62–68. IEEE Computer Society Press, October 1984.

[42] R.Nevatia and K.E. Price. Locating structures in aerial images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(5):247–255, 1982.

[43] M. Sasarman and D. Ionescu. A parallel watershed transformation for the detection of major road networks in landsat images. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, September 1995.

[44] D.L. Schanzer, G.W. Plunkett, and D.Wall. Filters for residential road delineation from spot pla imagery. In *Proceedings, GIS for the 1990s, Second National Conference on GIS*, pages 801–815, Ottawa, ON, Canada, March 1990. Canadian Institute of Surveying and Mapping.

[45] G. Smith and J. Austin. Analysing aerial photographs with adam. In *IJCNN International Joint Conference on Neural Netorks*, volume 3, pages 49–54, New York, NY, USA, June 1992. IEEE.

[46] P.J. Soille and M.M. Ansoult. Automated basin delineation from digital elevation models using mathematical morphology. *Signal Processing*, 20(122):171–182, 1990.

[47] P.J. Soille and L. Vincent. Determining watersheds in digital pictures via flooding simulations. *Proceedings of Visual Communications and Image Processing '90 - SPIE*, 1360(1):240–250, 1990.

[48] R. Swann, D. Hawkins, A. Westwell-Roper, and W. Johnstone. The potential for automated mapping from geocoded digital image data. *Photogrammetric Engineering and Remote Sensing*, 54:187–193, 1988.

[49] J. van Cleynenbreugel, F. Fierens, P. Suetens, and A. Oosterlinck. Delineating road structures on satelite imagery by a gis-guided technique. *Photogrammetric Engineering and Remote Sensing*, 56(6):893–898, 1990.

[50] G.J. Vanderbrug and A. Rosenfeld. Linear feature mapping. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-8:768–774, 1978.

[51] V. Venkateswar and R. Chellappa. Extraction of straight lines in aerial images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11):1111–1114, November 1992.

[52] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[53] F. Wang and R. Newkirk. A knowledge-based system for highway network extraction. *IEEE Transactions on Geoscience and Remote Sensing*, 26(5):525–531, 1988.

[54] J. Wang and P.J. Howarth. Methodology for automated road network extraction from landsat tm data. *Proceedings of the Ann ASPRS ACSM Convention (Baltimore, MD)*, 1:429–438, 1987.

[55] J. Wang and P.J. Howarth. Use of the hough transform in automated lineament detection. *IEEE Transactions on Geoscience and Remote Sensing*, 28(4):561–566, 1990.

[56] J. Wang, P.M. Treitz, and P.J. Howarth. Road network detection from spot imagery for updating geographical information systems in the rural-urban fringe. *International Journal of Geographical Information Systems*, 6(2):141–157, March-April 1992.

[57] B. Yee. An expert system for planimetric feature extraction. In M.C. Dobson, editor, *Proceedings IEEE International Geoscience and Remote Sensing Symposium*, volume 1, pages 321–326, Ann Arbor, MI, USA, 1987. University of Michigan.

[58] J.S. Zelek. Computer-aided linear planimetric feature extraction. *IEEE Transactions on Geoscience and Remote Sensing*, 28(4):567–572, 1990.

[59] A. Zlotnick and P.D. Carnine Jr. Finding road seeds in aerial images. *CVGIP: Image Understanding*, 57(2):243–60, March 1993.

Index