A Systemic Approach to Topographic Map Processing

A thesis submitted during 2011 to the University of Hyderabad in partial fulfillment of a Ph.D degree in Department of Computer and Information Sciences, School of Mathematics and Computer and Information Sciences

by

Sandhya Banda

Department of Computer and Information Sciences
School of Mathematics and Computer and Information Sciences

University of Hyderabad
(P.O.) Central University, Gachibowli
Hyderabad - 500 046
Andhra Pradesh
India

CERTIFICATE

This is to certify that the thesis entitled **"A Systemic Approach to Topographic Map Processing"** submitted by **Ms. B. Sandhya** bearing Reg. No **06MCPC05** in partial fulfillment of the requirements for the award of **Doctor of Philosophy in Computer Science** is a bonafide work carried out by her under our supervision and guidance. The thesis has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Prof. Arun Agarwal
(Supervisor)
Department of Computer and
Information Sciences,
University of Hyderabad,
Hyderabad.

Dr. Rajeev Wankar
(Supervisor)
Department of Computer and
Information Sciences,
University of Hyderabad,
Hyderabad.

Head of the Department

Dean of the School

DECLARATION

I, B. Sandhya hereby declare that this thesis entitled **"A Systemic Approach to Topographic Map Processing"** submitted by me under the guidance and supervision of **Prof. Arun Agarwal** and **Dr. Rajeev Wankar** is a bonafide research work. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree or diploma.

Date:

B. Sandhya

Regd. No.06MCPC05

Abstract

Topographic maps, commonly known as topomaps, are essential tools for spatial analysis with respect to various activities like infrastructure planning, construction, mining, disaster management and recovery, town planning, organization of roads and exploitation of natural resources. Contour lines in topomaps are curve features that connect contiguous points of the same terrain elevation (altitude) and convey three dimensional information of the terrain's surface. Vectorization of layer pertaining to contours, has a tremendous application to geo scientific modelling. Topomaps stored in paper form need to be converted into digital form for integration into geographic databases. Manual conversion of maps by an operator using a digitizing tablet, however, is time-consuming, costly, and error-prone. The development of a more efficient topomap capture and storage system is, therefore, of substantial importance. Topomaps have to be processed to separate various semantic overlapping layers which are graphical in nature. Extraction of semantic layers, raster to vector conversion, recognition of symbols and text, 3D modelling etc. are some of the research challenges involved in topomap processing which demand application of advanced image processing and vectorization algorithms. Time consuming vectorization process, presence of thin and closely spaced colour lines, existence of intersecting and overlapping layers, aliasing and false colours, textured background and sheer enormity in terms of document size are some of the topomap-specific features that contribute to the complexity of this research. The objective of this thesis is to identify and propose a systemic approach engineered on three significant stages in topomap processing, namely, Extraction, Representation and Reconstruction of contour layers of a topomap. Our endeavour is to provide an integrated solution, which

while solving all associated issues and reusing existing schemes, is well suited to implementation on parallel computation architecture. Our work is built on a solid foundation of extensive and selectively focused literature survey, the aim of which has been to identify the current state of the art in the field and leverage upon existing algorithms, credited with proven performance in handling similar image processing applications. The efficacy of the ideas proposed is well supported by results of simulation on real world data.

Acknowledgements

I would like to express my deep sense of gratitude to my research supervisors Prof. Arun Agarwal and Dr. Rajeev Wankar for their continuous guidance, support and encouragement in carrying out this work. I consider myself highly privileged to be mentored by Prof. Arun Agarwal whose vast knowledge and experience has immensely benefited me. I cannot forget the morale boosting effort of Dr. Rajeev Wankar on occasions when I lost hope. I express my sincere gratitude to Prof. C. Raghavendra Rao, Head, DCIS for having kept my motivation alive through out this endeavour and giving me valuable insights from time to time.

I thank Prof. T. Amaranath, Dean, School of MCIS for his support. I would also like to convey my sincere thanks to the members of DRC, Prof. Chakravarthy Bhagvati and Dr. Atul Negi who have periodically reviewed my work and have suggested new directions as and when required. I thank Prof. K. V. Subbarao, University Centre for Earth and Space Sciences, for introducing me to an interesting problem and providing me with input data.

I thank CMSD (Centre for Modelling Simulation and Design), for providing me the financial and logistic support. I also thank INCOIS (Indian National Centre for Ocean Information Services) for providing me with financial assistance at the right time.

I thank my fellow-researchers Uma Devi, Sapna, Mini, Rafah for sharing with me joys and problems.

I thank my father and mother who have always inspired me and allowed me to pursue my own dreams. I thank my husband, Satya, who was the first person to plant in my mind the idea of pursuing doctoral research and assuring me that I am capable of taking it up. I thank

my father-in-law and mother-in-law who have supported me at home in every possible way and encouraged me at every juncture.

Appreciation expressed in this acknowledgement would only belittle the contribution of my daughter, Shreya, who has let me steal from the precious time to be spent with her.

B. Sandhya

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols

A_g	Adjacency Matrix, of size $[Sum(V_g), Sum(V_g)]$
A_{rc}	Adjacency Matrix of size $[N_{rc}, N_{rc}]$
$Area_c$	Total object pixels in a component
Asp_c	Aspect ratio of components
$C, \{C_t, t = 1\ to\ N_{tc}\}$	Set of Clusters obtained from Algorithm 3.3
CI_{inp}	XYZ transformed image
cnt	2D Array of size $[N_{rc},\ N_{rc}]$
Den_c	Density of components
$dens\_img$	Density ratio of image
$Epts$	End points of gap segments
F	3D matrix of size $[N_r,\ N_c,\ 18]$
$GCIm$	Gap Contour Image
$GCTmp$	Gap Contour image in a window
GI_s	Set of images which store gap segments
HI_{inp}	HSV transformed image
i, j, k	Loop indices
I_q	Quantized Image of size $[N_r,\ N_c,\ 3]$
I_f	PCA transformed image of size $[N_r,\ N_c,\ 3]$
I_{inp}	RGB topomap image, 3D matrix of size $[N_r,\ N_c,\ 3]$
I_S	$\{I_S^C, C = 1\ to\ N_{cc}\}$ Set of connected components
Img_c	Contour layer of topomap
Img_s	Input contour image(binary)
Img_w	Portion of contour image in a window
Img_{f1}	Contour layer image after filtering Type 1 tags
Img_{f2}	Contour Image after filtering altitude tags
Img_l	Contour line (component of I_{f2} obtained from 3.6)
L	Set $\{L_i, i = 1\ to\ N_l\}$ of leaf nodes
$L, \{L_t, t = 1\ to\ N_l\}$	Merged layers of topomap
l_c, h_c	Length and height of MBR_c
L_i^I	Set of leaf nodes intersecting with node L_i
L_{t1}	Lists of components belonging to text
L_{t2}	Lists of components belonging to text and contours
LI_{inp}	CIE $(L^*a^*b^*)$ transformed image
M_S	MBR data of segment
$MArea_c$	Area of MBR's of components
MBR_c	MBR of a component
N	Size of sub images
N^I	Set of leaf nodes intersecting with node N

N_g	Number of windows consisting of gap segments
N_l	Number of layers
N_w	Number of windows/sub images in Img_s
N_{cc}	Number of connected components
N_{gap}	Number of gaps
n_{ij}^p	Number of pixels of cluster RC_j in cluster RC_i in a $(w,\ w)$ neighbourhood
N_{img}	The size of image Img_s
n_{pc}	Count
N_{rc}	Number of clusters after removing noise, background
N_{tc}	Number of clusters
n_T	Pointer to a structure denoting node of a tree (Ref: Figure 4.3)
Q_C	Quantization levels along 3 axes, Set of 3 values
$RC,\ \{RC_t, t = 1\ to\ N_{rc}\}$	Clusters obtained after removing noise, background
S	List of nodes required to be traversed in order to trace the curve
S^z	List of size N_{tc}
S_M^z	Maximum cluster size
$strow,\ stcol$	Starting row,column of window in Img_s
T	Threshold on density ratio
T_1	0.0001 (Threshold to remove noise)
T_2	Threshold
T_{Asp}	Threshold on Aspect ratio of components
T_{Den}	Threshold on Density of components
T_{MArea}	Threshold on MBR Area of components
T_{pc}	Threshold
V_g	Vector storing number of gap segments in each window
W	Set of Gap contour images
w	Size of window
w_g	Size of window (Ref: Section 5.1.1, Step i)
w_l, w_h	Window length and height
X	Feature set, 2D matrix of size $[N_r \times N_c,\ 18]$
Y	Array of size $[N_r \times N_c,\ 3]$
$Y'I_{inp}$	YIQ transformed image
YI_{inp}	YC_bC_r transformed image

Chapter 1

Introduction

Automatic spatial feature extraction and recognition of cartographic documents for data capture of Geographical Information System (GIS), represents a research challenge at the intersection of GIS and Image processing. An informal definition of GIS has been given as (D89): "...a system of hardware, software and procedures designed to support the capture, management, manipulation, analysis, modelling, and display of spatially referenced data for solving complex planning and management problems". GIS is now a gateway for accessing and integrating geographic data from different sources located locally and globally. It is being increasingly used for interactive visualization of scenarios resulting from different business decisions, as well as for the communication of spatial knowledge and intelligence among people all over the world (AK05).

A fundamental characteristic of a GIS is its ability to handle spatial data, i.e. locations of objects in geographical space, and the associated attributes. The first step in the building of a GIS is data acquisition. A map, which is a graphical representation of the spatial structure of the physical and cultural environments, is one of the primary sources of data in a GIS. Since computer assistance of geographical data processing has become popular, there is a latent need to produce new digital maps and to convert old analog ones for integration into digital systems, like GIS (LMR96). There are two categories of maps based on the purpose of use:

1. General-purpose: This class of maps focuses on locations and show a variety

of physical and cultural features such as landforms, roads, railways, airports, built-up areas, forests and cultivated areas. Ex: Topographic maps

2. Special purpose: Thematic maps which are designed to depict a particular type of feature or measurement only like population distribution, rainfall distribution etc.

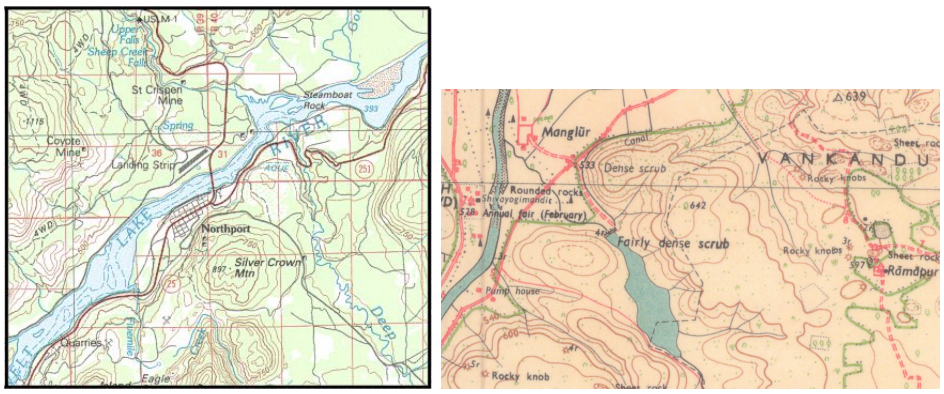
This work relates to a class of topographic maps that describe the terrain features which fall under category 1.

1.1 Topographic Maps

A topographic map gives a detailed and accurate graphic representation of cultural and natural features of the terrain based on topological surveys (AK05). Topographic maps commonly known as topomaps are essential tools for spatial analysis with respect to various activities like infrastructure planning, construction, mining, disaster management and recovery, town planning, organization of roads and exploitation of natural resources. The information on these maps consists of sets of linear features and area features. Linear features are lines on the topographic map that convey geographic information about the locations of rivers, roads, and political boundaries, whereas, area features are regions on the topographic map that portray information about the location and placement of buildings, vegetation, and bodies of water. Topomaps describe spatial data using entities, attributes and relationships in an unstructured representation. Entities on maps are described by point, line, and area objects. Symbols, Text, Colour are commonly used on maps to represent attributes associated with the entities. Relationships among entities are depicted visually.

1.1.1 Standards of Topomaps

Topographic maps are available from many suppliers, each supplier specifying standards for linear and area feature attribute. These standards specify everything from the colour and thickness of a linear feature to the shape and colour of an area feature. One of the largest suppliers of topographic maps is the United



(a) Sample of USGS topomap (From (Map11))

(b) Sample of SOI topomap

Figure 1.1: Sample Topomaps

States Geological Survey (USGS), available for the entire territory of the United States in several map scales from 1:24,000 to 1:500,000. In India, topomaps of the regions conform to the standards of Survey Of India (SOI). A sample 1:100K USGS topomap is shown in Figure 1.1(a) and a map (part of 1:50000, 48M/13 First Edition) prepared according to SOI standards, is shown in Figure 1.1(b). The Figure 1.2 shows the standard symbols and colours used for SOI topomaps.

Topographic maps use very large map scales and very thin feature geometry to convey as much terrain detail as possible. Because of the large number of measurements involved, spatial analysis based on maps has been a very time-consuming and tedious task.

1.2 Contour Layer

The most distinguishing feature on the topographic map are contour lines, which show the elevation of the terrain. Contour lines are curve features that connect contiguous points of the same terrain elevation (altitude) i.e. all the points on a contour line have the same elevation. They convey three dimensional information of the terrain's surface. It is important to note that without these contour lines, a topographic map degenerates to a planimetric map providing no three-dimensional data about the terrain.

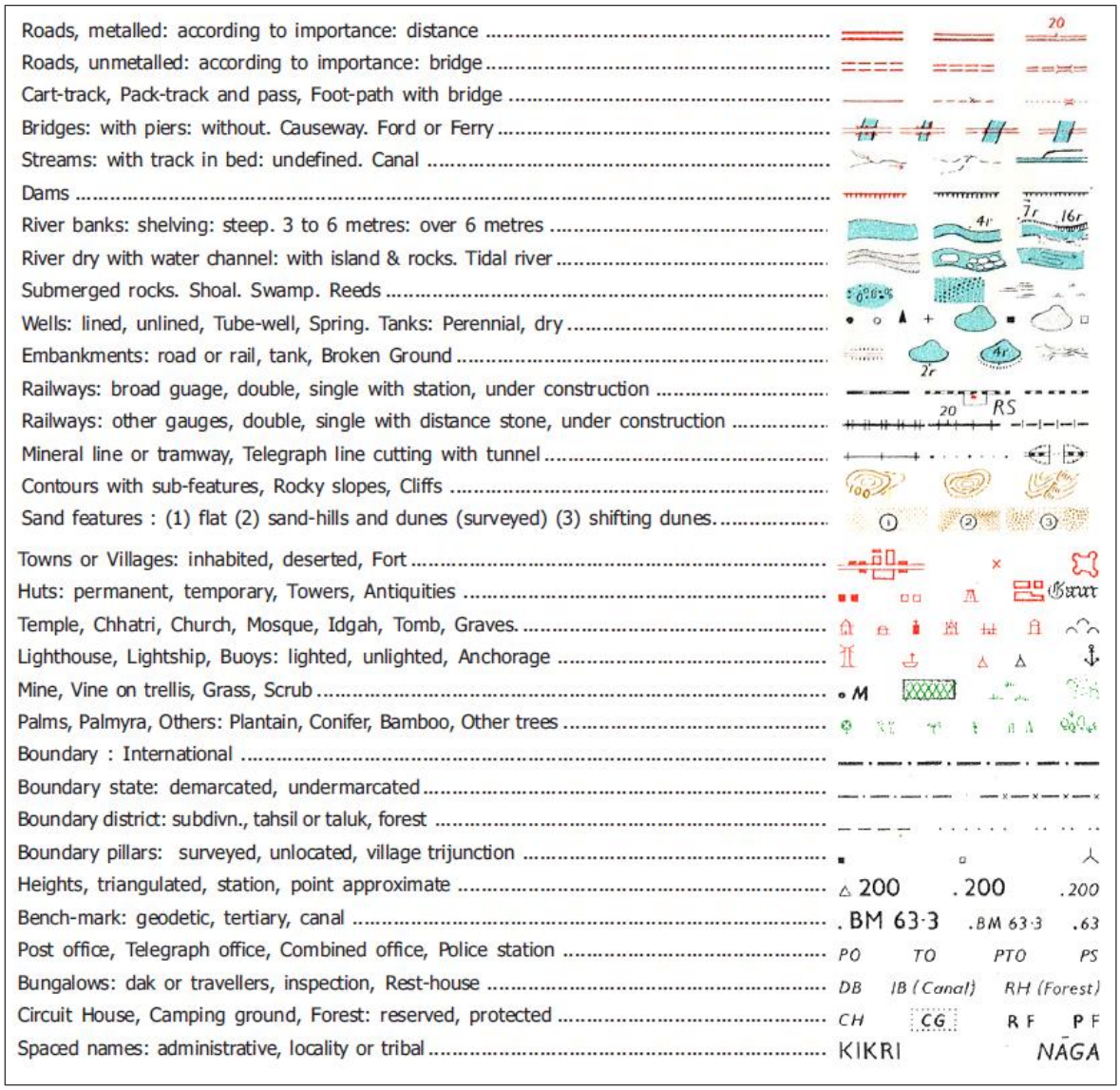


Figure 1.2: Symbols of Topomap (Source: Survey of India)

(a) Index contour (b) Intermediate contour (c) Supplementary contour

Figure 1.3: Contours of Topographic Map

Three types of contour lines are found on a topographic map. The index contour lines are dark brown contour lines with their elevation written on them, shown in Figure 1.3(a). Between index contour lines are light brown lines drawn at regular intervals, known as intermediate contours, shown in Figure 1.3(b). These lines have no numbers. Supplementary contour lines are occasionally used and are used to denote changes in elevation between intermediate contour lines. Supplementary contour lines are drawn as dashed brown lines, as shown in Figure 1.3(c). These dotted lines are placed in areas where elevation change is minimal. If there is a lot of space between Index and Intermediate Contours (as happens where the land is relatively flat), Supplementary lines are added to indicate that there are elevation measurements, even if they are few and far between. These contour lines allow the elevation of any point on a topographic map to be estimated.

The topological characteristics of contour lines are, i) they are continuous, i.e contour lines are either closed or end at the boundaries of the map. ii) Contour lines do not intersect each other. Contour lines are represented by lines with 2-4 pixels width when scanned at a resolution of 300dpi, and are brown in colour. The lines are sometimes labelled indicating the elevation values above the mean sea level. However the characteristics of contour lines are not always preserved when extracted from a topographic map.

1.3 Motivation

Topomaps are unique data sources for GIS based analysis of land cover changes over long time periods, and typically are the only spatially contiguous data sources prior to the establishment of remote sensing. Large amounts of historical maps are currently collected in archives and become available as scanned data sources.

These old topomaps can be of great relevance (and justifies further research), where present data could be compared with the topomaps generated over the years. For example, a lake shown in old topomaps may be missing in present context. To make use of such maps, automatic approaches for spatial feature extraction and recognition are necessary.

The objective of an automatic map interpretation system is to generate a symbolic description for all map entities and their spatial relationships, which is then stored in a geographic database. Pattern recognition in cartographic documents aims at the delineation and extraction of spatial information and its incorporation into GIS. However to establish methods for recognition in maps is particularly challenging due to:

- Complexity of map contents: Though certain standards are employed to represent the various contents of a map, the directions and sizes of individual elements differ.

- Various elements of maps overlay and intersect mutually.

Watanabe (Wat00) has categorized various kinds of drawings in terms of recognition complexity using individuality of composite elements and correspondence among them as the main criteria. Figure 1.4 shows the categorization. It can be observed from the Figure 1.4 that, maps are located in the right-upper side, where the individuality and correspondence are relatively lower than those in the other drawings. This makes recognition of individual elements (or objects as single recognition entities) more difficult.

Extraction of semantic layers, raster to vector conversion, recognition of symbols and text, 3D modelling etc. are some of the research challenges involved in topomaps processing which demand application of advanced image processing and vectorization algorithms.

The following issues motivate research in the area of automatic topomap processing:

- Vectorization of Paper-based topographic maps is tedious and time-consuming process as it involves lot of manual effort.

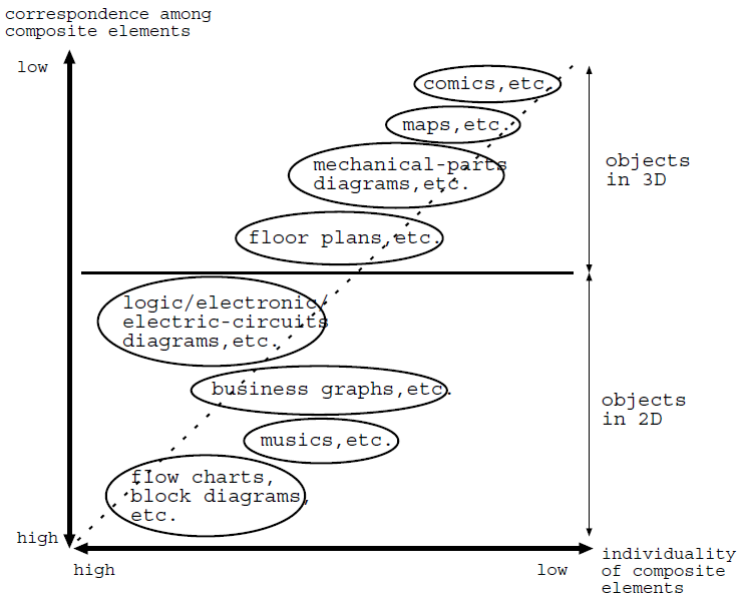


Figure 1.4: Feature of Drawings (From (Wat00))

- Not many of the existing techniques are focused on colour line drawings with thin closely spaced linear features.
- Current techniques of fully automatic vectorization for topographic maps are not efficient in extracting contour lines, due to the inherent features of topomaps described below.
- Processing large topomaps require enormously high execution time. Parallel programming has to be considered to obtain viable solution in reasonable time.

1.3.1 Document Image Analysis

Topographic maps stored in paper form need to be converted into digital form for integration into geographic databases. Manual conversion of maps by an operator using a digitizing tablet, however, is time-consuming, costly, and error-prone. The development of a more efficient map input system is, therefore of substantial importance. As topomaps belong to the class of paper documents, their analysis is considered as a branch of Document Image analysis. Therefore major steps to

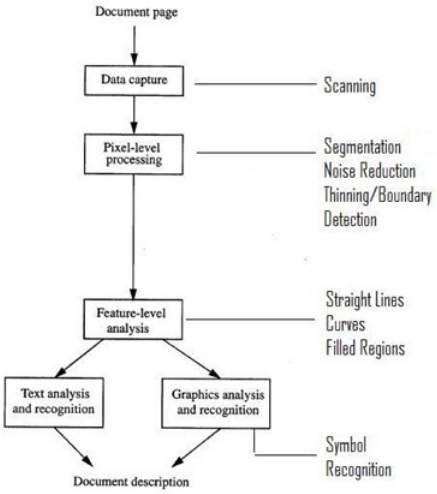


Figure 1.5: A Typical Sequence of Steps for Document Analysis (From (KGG02))

be followed include data capture, pixel level analysis and feature level analysis, as shown in Figure 1.5 (KGG02). However unlike many other documents where the focus is on separating text from graphics, topomaps have to be processed to separate various semantic overlapping layers which are graphical in nature.

Data capture is done with the use of a scanner as input device, and the scanned image is subjected to further image analysis techniques, to automatically convert a paper-based map into atomic objects that can be interpreted and processed by a computer. At a typical sampling resolution of 120 pixels per centimetre, a 20×30 cm map would yield an image of 2400×3600 pixels. Scanning at 300dpi resolution is a compromise in order to achieve all the elevation details from the printed map while avoiding too many printing details (RA05).

The next step in document analysis is to perform processing on the captured image to prepare it for further analysis. Such processing includes: Segmentation to separate various components in the image, Thresholding to reduce a grayscale or colour image to a binary image, reduction of noise to reduce extraneous data, and, finally, thinning or boundary detection to enable easier subsequent detection of pertinent features and objects of interest. After such processing data are often represented in compact form such as chain-codes and vectors. In the case of topomaps steps under this pre processing include segmentation of topomap to

separate various layers, filtering of layers to remove noise and finally representation of the layers.

After pixel-level processing on the document image, intermediate features are found from the resulting image to aid in the final step of recognition. At the feature level, thinned and chain-coded data is analysed to detect straight lines, curves and significant points along the curves. For contour layer, this stage includes processing steps such as approximation of contour segments, identification of end points in the contour segments which will further assist in gap filling.

The methods of analysis of linear features in other types of documents (e.g. engineering drawings), which have been reported in the literature do not take advantage of the specific map knowledge. A significant feature of maps is that the road networks, iso-contours, etc. offer a more extensive and coherent linear network than other types of documents (HL09).

1.3.2 Challenges in Topomap Processing

Topographic maps contain great amounts of information superimposed onto a single 2D layer, resulting in a complex mixture of touching text and graphics components. Automatic extraction of different layers of information poses a substantial challenge due to heavy interconnection of these layers. Additional challenges are as follows:

- Aliasing and Existence of False colours: Introduced due to scanning process. The aliasing induced by the scanner's point spread function results in smoothing of the edges between foreground and background layers. Another problem is the introduction of false colours due to lateral chromatic aberration which in turn is a result of the difference between the refractive index of the scanner's lens for different wavelengths of light. For example in a simple scanned USGS map consisting only of blue, brown, black and purple, over 42000 distinct colours can be detected including some brilliant shades of green, yellow, and violet which are neither visible to the naked eye nor exist in the original map (PT08).

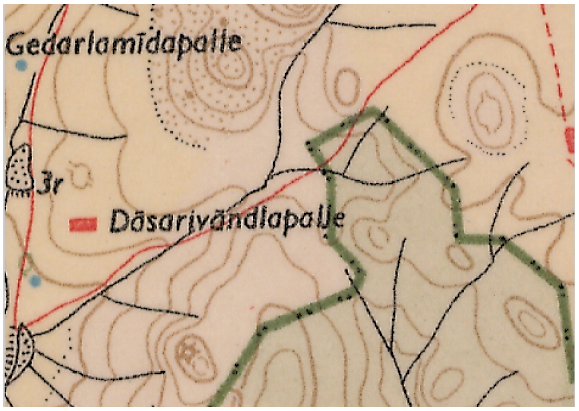


Figure 1.6: Textured Background

- Closely spaced features: When two features are closely spaced, the background separating them is eroded by the scanner induced aliasing to the point where it is difficult to use background to split these features.

- Intersecting and overlapping linear features: Topographic map is cluttered with data like contour lines, trails, streams, vertical coordinate grid line, etc. Separating each of these features can cause one or both of them to appear broken.

- Textured Background: Different background colours and textured patterns are used to represent features such as bodies of water and vegetation. This increases the complexity of Extraction of layers as the colour of linear features changes in contrast to their background. For example in the Figure 1.6 , which is part of the topomap 1:50000, 57 J/4 First Edition, SOI, the area of forests is shown in green and the contour lines which pass through the green areas have different shades as compared to other contour lines.

- Extremely large processing times due to large size topomaps

Current approaches in recognizing contour lines from scanned colour topomap are insufficient as they are either time consuming or involve much human effort. Hence an algorithm which is fully automatic and less computationally intensive should be developed to extract the contour layer and to eliminate undesired features. In addition algorithms to detect and fill the discontinuities in contour

lines caused due to overlapping/intersecting features of topomap, or improper scanning/extraction process have to be proposed. Since the goal of topomap processing is its incorporation into a GIS for land cover change detection, generation of a 3D model etc. a suitable representation scheme for contour layer is essential which can efficiently handle spatial operations at a region level and pixel level.

1.4 Major Contributions

We now briefly discuss our main contributions made in the processing of topomaps:

- **Extensive Survey**: Topomap Processing is a multi-disciplinary field drawing contributions from several areas like Geo Science, Image Processing, Pattern Recognition, Computer Vision and Parallel Computing. The complexity arises from the fact that the processing involves a large number of inter-dependent stages. Consequently research in the field has witnessed significant contributions with respect to all aspects of the problem. Extensive literature survey was needed as an essential pre cursor to understand the depths and intricacies involved so as to propose an integrated solution to the entire problem.

- **Contour layer Extraction**: Binary image of contour layer is obtained from a topomap by employing colour clustering algorithms. The clustering is performed with colour features extracted through a feature selection approach. The resulting clusters are post processed using neighbourhood analysis to obtain the meaningful layers of topomap.

- **Filtering of Contour layer:** The resultant contour layer contains altitude values in addition to the contours, as they are represented using the same colour as contours. The proposed filtering method helps to produce a clean contour layer free from elevation values.

- **Representation of Contour layer:** A hierarchical scheme of representing the contour layer of a topomap is proposed which can handle the various operations on the contour lines at a region level or at a pixel level.

- **Reconstruction of Contour layer:** A novel hybridized algorithm is developed for reconstructing the extracted contour lines from colour topographic map. This algorithm is based on connected components, Expectation Maximization (EM) and numerical methods. A new scheme of filling gaps, present in thick contours, without the application of thinning algorithms is presented.

Organization of the Thesis

Our work is organized into seven chapters as follows:

Chapter 1. introduces topographic maps and their application in the area of Geo Science and motivation for pursuing research in this area.

Chapter 2. reviews the research done in the various areas of topomap processing.

Chapter 3. deals with the Extraction of contour layer which includes Feature selection, Clustering, Separation and Recognition of altitude tags.

Chapter 4. proposes a new scheme for representation of the contour layer and briefly discusses how spatial queries are handled.

Chapter 5. explains the proposed Gap identification i.e identifying the broken segments of contour line and matching the segments which should be joined to form a closed contour. The chapter also includes implementation details of Gap Identification algorithm on a parallel architecture.

Chapter 6. focuses on the various operations like Approximating contour segments, Gap filling and tagging recognized altitude values to the contour lines.

Chapter 7. concludes the thesis by summing up the contributions made and discusses future scope of research in this area.

Chapter 2

Literature Survey

The recognition of maps and geographic documents has been researched as an interesting and difficult subject for composition of resource data in the construction of GIS (TC98). This has been mainly done from the perspective of automatic extraction of map composite elements. Graphical images have colour, geometric (location, shape), topological and attributive (quantitative and qualitative parameters, e.g. the name of object) information, which we can merge into the concept of cartographic image.

Research on fully automatic feature extraction and object recognition from scanned topographic maps deals with two main difficulties:

1. Automatic extraction of layers
2. Accurate recognition

The focus of our research has been automatic extraction and vectorization of contour layer from colour topographic map. Hence the survey highlights research publications relevant to contour layer extraction and vectorization.

2.1 Extraction/Segmentation of Layers

The scanned topographic map is stored as a digital colour image in RGB space. Colour is the prominent feature to distinguish between various thematic layers of

the map, such as hydrography, elevation contours, vegetation etc. Hence colour segmentation techniques are used to separate the various layers of topomap.

The majority of colour segmentation approaches are based on monochrome segmentation approaches operating in different colour spaces. All the existing colour segmentation algorithms are strongly application dependent, in other words, there are no general algorithms and colour space that are good for all colour images (CJSW01). There are two critical issues for segmentation of topographic maps:

1. What segmentation method should be utilized?

2. What colour space should be adopted?

The segmentation approach can be histogram thresholding, clustering, region growing, edge detection, fuzzy approaches, neural networks or a combination. Several colour spaces, such as RGB, non linear colour spaces like HSI, $CIE(L^*a^*b^*)$, $CIE(L^*u^*v^*)$, linear colour spaces like YIQ, YIV, $I_1I_2I_3$ etc. are utilized in colour image segmentation. However none of the colour spaces can be preferred over others for all kinds of colour images. Selecting the best colour space still is one of the difficulties in colour image segmentation (CJSW01).

Inherent characteristics of topomaps pose several challenges to colour based separation of features. These are further accentuated by scanning, ageing as explained in the previous chapter. Hence a variety of pre processing techniques are employed to overcome the problems of false colours etc. The contour layer obtained from segmentation contains artifacts and noise and hence post processing techniques to filter are necessary for generating a clean contour layer. Table 2.1 lists the approaches followed in some of the publications.

The method proposed by (LB10) uses a seeded region growing (SRG) process using colour similarity and spatial connectivity. However the colour layer prototypes for SRG are derived based on certain assumptions and parameters like window size for local homogeneity, thresholds for measuring frequency, colour similarity etc. (PT08) et al. use CLAHE (Contrast Limited Adaptive Histogram Equalization) method for pre-processing. Segmentation is based on thresholding of distance from the sample colour patch which is selected by the user. Hence

it is only a semi-automatic method. In (RBO07) Genetic algorithms are used to construct an Adaptive Colour Space (ACS). The fitness is based on the colour recognition rate given by the supervised region growing segmentation algorithm. In (RRA04) methods based on colour image processing like vector edge detector, vector median filter, minimum variance quantization are used for noise filtering, removal of isolated linework pixels, segmentation respectively. The method is computationally expensive and is not fully automatic. An approach to extract land-use classes from colour thematic maps is presented in (Cen98). Colour segmentation based on histogram thresholding of $I_1I_2I_3$ is used. Further, Delaunay triangulation is employed and spatial segmentation is done based on the perimeter of the triangle. The method is sensitive to noise and fixation of threshold values is still a human task and should be improved.

As can be observed, there is no particular method or colour space which can be employed efficiently on topographic maps generated using different standards or scales. For example the colour key set developed by (KZ03) can be used for USGS based topographic maps only. Also most of the methods require human intervention and are not fully automatic (HL09; CWQ06a). Further the results of algorithms are sensitive to the pre processing employed to clean the map as in the case of (RM00) where results are affected due to quantization.

Accuracy assessment of extraction are rarely done according to literature (CWQ06a) or they are focused on line segmentation taking into account the line length as evaluation measure (CWQ06a; KZ03). A method for objectively assessing the accuracy of segmentation results where area, line and symbol regions occur is presented using confusion matrix based assessment in (LB10), which is used in the present work also.

2.2 Vectorization of Contour Layer

Significant research has been carried out specifically on extraction and recognition of contour lines. However algorithms reported in the literature performed well on the binary images of maps that have thicker features and features that are spaced far apart from each other (HCY08).

Table 2.1: Extraction of Layers from Colour Maps - Survey

S.No	Paper	Pre processing			Segmentation			Post processing
		Objective	Method	Input	Features	Method	Inputs	Method
1	(LB10)	Finding prototypes of colour layer	Clustering local homogeneity, Peak finding based on frequency and similarity	Window size for local homogeneity, Threshold for measuring frequency, Radius of sphere for measuring similarity, Minimum frequency of same colour pixels	RGB	Seeded region growing	–	Filter unallocated pixels using an approach similar to median filtering
2	(HL09)	-	-	-	RGB	Thresholding of colour and histogram count	Colour and frequency threshold values	Tensor voting method to extract linear features
3	(PT08)	Quantization	By contraction and expansion of histogram of L values	CIE Lab, Grid size and clip limit in CLAHE method of histogram expansion	RGB	Threshold on distance from selected region	The region to which distance is measured	-

Continued on Next Page...

Table 2.1 – Continued

S.No	Paper	Pre processing			Segmentation			Post processing
		Objective	Method	Input	Features	Method	Inputs	Method
4	(RBO07)	Image restoration(ancient cadastral maps)	White patch and faded colour correction	-	Hybrid colour space (HCS) computed using GA	Region growing segmentation	Training data to obtain HCS	-
5	(CWQ06a)	To find colour key set of map	-	-	RGB	Clustering using histogram of RGB	-	local window technique on a Gray image
6	(RRA04)	1. Noise removal filtering 2.Linework removal	Wavelet domain hidden Markov model denoising on each plane Vector edge detector	RGB Saturation based combination of hue	RGB	Minimum variance quantization	Number of regions within the image + 1	Gap filling using Wise algorithm

Continued on Next Page...

Table 2.1 – Continued

S.No	Paper	Pre processing			Segmentation			Post processing
		Objective	Method	Input	Features	Method	Inputs	Method
		3.Remove isolated linework pixels	Vector median filter	and intensity planes				
7	(KZ03)	Construction of colour key (to overcome aliasing and false colours)	Solid colour keys by computing mean Combination colour keys: From end points of the ellipsoid patterns formed by the pixels	Based on user input of 16 x 16 pixel samples 5 samples for each defined colour key	RGB	Connected solid colours: Colour clustering of histogram of RGB Linear feature: Valley seeking algorithm	Colour keys	extract contour lines by constructing local minimum adjacency graph , Gap filling by A* search algorithm
8	(RM00)	Quantization	Uniform	RGB	RGB	k-means clustering into GA	Number of clusters	
9	(Cen98)	Colour space conversion	KL transform	RGB	$I_1I_2I_3$(of KL)	Thresholds from 1D histograms		Delaunay triangulation, threshold on perimeter of triangle

Continued on Next Page...

Table 2.1 – Continued

S.No	Paper	Pre processing			Segmentation			Post processing
		Objective	Method	Input	Features	Method	Inputs	Method
						Merging clusters using euclidean distances bet cluster centres		Morphological operations
10	(ELA94)	Transform RGB	To L*u*v* colour space	-	L*u*v*	Maxima of u*v* histogram, Distance from additive colour mixture lines and threshold on chromaticity	Thresholds on chromaticity, distances	-

Early methods proposed for vectorization of line drawings necessitated the following steps to facilitate automation:

1. topographic map digitization by scanner
2. thresholding
3. thinning and pruning the resulting binary image
4. raster to vector conversion of the resulting thinned lines.

Automatic vectorization of clean contour and drainage/ridge sheets has been addressed in (FD82). One of the initial efforts to extract elevation contour lines on topographic maps is presented in (Gre87). The above steps have been also used in (TR87) for the recognition of lines and symbols, and (MMEA88) for the processing of land record maps (AS99). Since colour maps are now increasingly being used for processing, thresholding alone is not sufficient to obtain a binary image. Hence the second step of processing is modified to, colour image segmentation and filtering noisy pixels, which has been discussed in the previous section. Therefore, references mentioned in this section exclusively point to the problem of raster to vector conversion which has been dealt with in one of the following ways (SG04; JS07):

1. Image based approach
2. Geometric based approach
3. Gradient vector flow approach

Image based Approach

In this approach of raster to vector conversion, closing or grouping two different segments/pixels is strictly based on perceptual principles i.e using the two main criteria, proximity and continuity. (LKH95) used line tracing algorithm technique for contour line reconstruction. The problem of gap reconstruction is solved by assuming that there is only one possible continuation from an end-point. The natural continuation can be found along the current direction of the line. The

gap is crossed by searching from the point at the end of the line within a sector around the current direction. This approach is used in (CWQ06b). In (AS99), contour lines from colour topographical maps are reconstructed using techniques based on mathematical morphology. A combination of Euclidean distances between extremities and differences between their tangential directions is used to join the disconnected lines in a very local fashion. This approach is used in (KZ03; HCY08) coupled to a A* search algorithm. In (RA05), in order to fill in the gaps a multi-pass majority filter - the Wises algorithm (Wis95) is run on the colour-clustered image to reclassify the white pixels on the basis of their surrounding pixels. In (CWQ06a) a local window segmentation technique based on the line-tracing algorithm is used at the gaps and thick lines, and global properties of contour lines are used to filter the accurate contour lines. In (RIC10), terminal points are found by scanning the thinned image and then directional relationships, Euclidean distance information are used for matching terminal points and connecting broken line segments.

Albeit attractive because of their simplicity, almost all the existing closing algorithm based on perception criteria fail at discontinuity points. .

Geometric based Approach

The curve reconstruction problem is analysed as an instance of the more general problem: given a finite sample V of an unknown curve λ, the task is to construct a graph $G = (V, E)$ in such a way that two points in V are connected by an edge of G iff the points are adjacent on λ. The graph G is called a polygonal reconstruction of λ. The curve reconstruction problem has received a lot of attention in the graphics and the computational geometry community and a great amount of work has been written. The first algorithms for curve reconstruction imposed a uniform sampling condition, as they basically demanded that the distance between any two adjacent samples must be less than a given constant. This is not satisfactory as it may require a dense sampling in areas where a sparse sampling is sufficient.

Amenta et al. (ABE98) defined the non-uniform sampling condition using the concept of the local feature size (distance of a point to the medial axis of his curve). A point in the plane belongs to the medial axis of a curve, if at

least two points located on the curve are at the same distance from this point. This algorithm works by computing the Delaunay Triangulation of the point set and then filtering it to obtain the reconstruction. Later, algorithms have been proposed based on Delaunay Filtering which can handle single closed curves (DRM99; Gol99). Though in (DW00) an algorithm that handles corners and endpoints has been proposed, it is not difficult to find counterexamples where it fails. In (SG04), authors propose to vectorize the contour lines using a Delaunay triangulation where the Delaunay edges are filtered using both local and global rules. Recently (GB11) also uses the reconstruction method proposed by (ABE98).

Gradient Vector Flow Approach

Zhou and Zhen (XH04) have proposed deformable model and field flow orientation method for extracting contour lines. An application to snakes is given in (DP05). Dongjun et al. (XZZ06) has suggested a method based on Generalized Gradient Vector Flow (GGVF) snake model. After thinning the contour lines, they extract a set of key points using the c-means algorithm. These key points become input in the GGVF snake model used for extracting the curves. In (JS07), reconstruction of contour lines is done based on the gradient orientation field generated by the input contour lines. In (PKA$^+$09), authors have extended the work of Dongjun et al. (XZZ06) to trace the contour lines more efficiently and automatically using Modified Moores Neighbor tracing algorithm. The use of Thin Plate Spline interpolation techniques are suggested in (AM09).

Table 2.2 summarizes the publications which have addressed both Extraction and Reconstruction of contour lines. As can be observed from the table, topomap processing is a multi-stage process and involves combination of colour space selection, segmentation, filtering and reconstruction approaches. However a fully automatic method which can be applied on any topographic map to extract a clean contour layer and further reconstruct it has not been reported. Most of the methods have constraints in terms of quality of digital cartographic document, processing requirements, manual intervention. They are either time consuming

and require heavy computations using powerful workstation or involve much human effort in assisting or correcting recognition error.

2.3 Our Approach: Components of Topographic Map Processing

Figure 2.1 is our visualization of various stages and the information flow in topomap processing. Though work has been done in the area of topographic map processing, research work which presents a systemic approach to the problem taking a colour topographic map as input, and giving a reconstructed contour layer along with a representation scheme as output has not been reported. Our thesis contribution is in the areas of Extraction, Contour line Representation and Reconstruction and is summarized in Figure 2.2. We briefly discuss each of our stages for topomap processing:

Stage I: As mentioned earlier, a topomap consists of various features differentiated primarily based on colour. Hence clustering algorithms based on colour are applied on topomap to extract each of these features into a separate layer. The objective of the Stage I is to separate the contour layer from various other features of topomaps like water bodies, vegetation etc. Prior to applying clustering on a topomap, the colours are quantized such that the original colours of the image are mapped to a smaller subset of colours. Clustering is followed by merging of the original clusters which further helps in extraction of meaningful layers from the topomap. The topomap is thus segmented into various layers based on their colour features, for example a layer of water bodies (in blue), a layer of text and grid lines (in black), a layer of contour lines (in brown) etc.

The resulting contour layer is not clean, and consists of artifacts i.e some pixels of other layers not belonging to contour lines, due to problems of aliasing and false colours in a colour topographic map. Apart from contour lines and artifacts, the contour layer also contains the altitude values which are represented using the same colour as contour lines. Hence algorithms for filtering the artifacts and for separation of text from graphic images are essential.

Table 2.2: Survey of Contour Layer Digitization

Paper	Extraction		Filtering	Reconstruction	Remarks
	Col space	Segmentation			
(GB11)	CIE L*a*b*	k-mean clustering	morphological filters, thin, prune	Geometric	Fails if map is poor
(RIC10)	RGB	Thresholding	Morphological Operations, thin, Median filter	Image	Heuristic approach
(PKA$^+$09)	HSV	Thresholoding	thin	Gradient	
(HCY08)	RGB	combines spatial and colour information	Morphological filters, thin	Image	feature set developed based on Gauss kernel function
(XZZ06)	HSI	Thresholding	Morphological filters, thin	Gradient	Uses c-means to find points on contours:not accurate
(CWQ06b)	RGB	Colour Histogram analysis	morphological filters, thin, prune	Image	
(RRA04)	HSV	Vector angle edge detector	median filtering, minimum variance quantization	Image	computational expensive and not fully automatic
(JY04)	HSI	Thresholding	median filter, thin	Geometric	Works if contour lines are not too fragmentary
(SYSbNI04)	HSV	Thresholding	thin	Image	
(SG04)	HSV	Thresholding	thin, smoothing	Geometric	User interaction necessary to solve problems
(KZ03)	RGB	Colour Histogram analysis	-	Image	Colour key set for map should be constructed
(AS99)	RGB	Thresholding	Morphological hit-miss transform	Image	Poor reconstruction

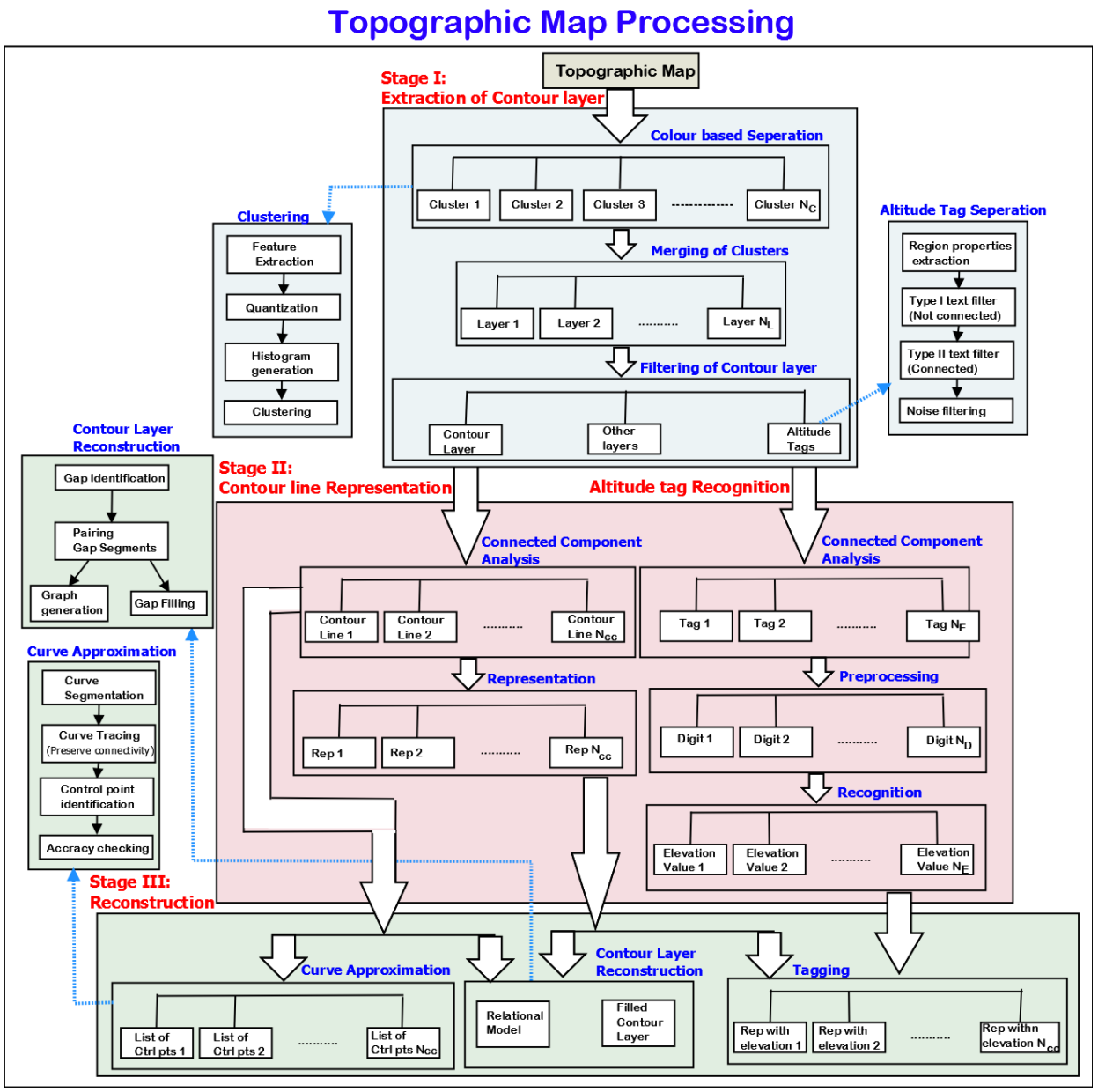


Figure 2.1: Topographic Map Processing

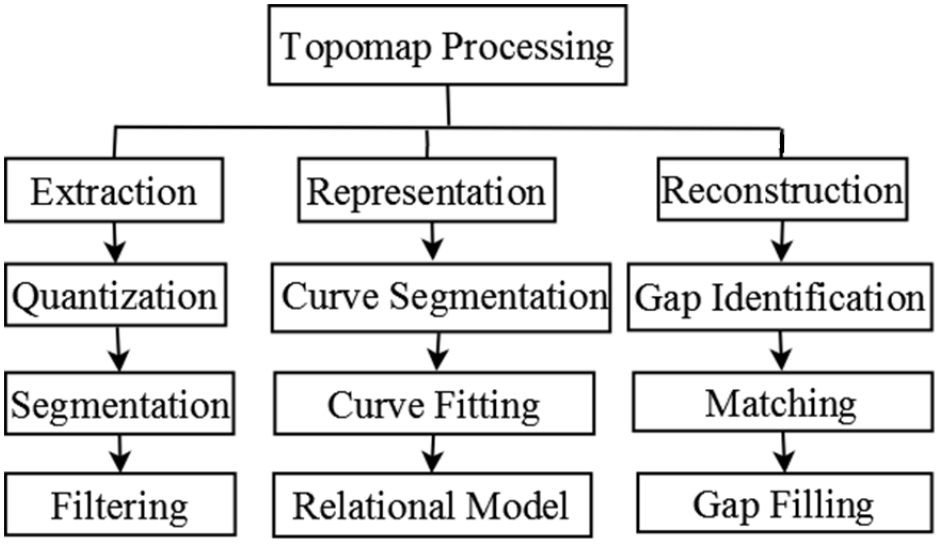


Figure 2.2: Stages of Topomap Processing

Stage II: After the Stage I, a clean contour layer, consisting of segments of contour lines is extracted from the topomap. The filtered altitude tags form another layer and the rest of the information of topomap is stored in 'other layer' as it is useful in later processing steps. Each contour line from the isolated contour layer is extracted and represented using curve representation techniques. Similarly altitude tags are segmented and recognized. This forms the second stage of topomap processing which outputs a contour line representation.

Stage III: Extraction stage leads to gaps in contour lines due to intersecting and overlapping features. So prior to vectorizing the contour layer, it becomes essential to fill the gaps. The third stage is Reconstruction, which involves the algorithms for identifying contours which contribute to a gap, pairing contour segments which form a gap and then filling the gaps. This stage takes a binary contour image as input and generates a reconstructed binary contour image as output. The recognized elevation values are tagged to the contour lines and an attribute which stores the altitude value is updated in the representation of contour line.

Chapter 3

Extraction of Contour Layer

A topographic map gives a detailed and accurate graphic representation of cultural and natural features of the terrain based on topological surveys (AK05). Topomaps also describe spatial data using entities, attributes and relationships in an unstructured representation. Entities on maps are described by point, line, and area objects. Symbols, Text, Colour are commonly used on maps to represent attributes associated with the entities. Relationships among entities are depicted visually.

A scanned topographic map, which is in raster form, is one of the primary sources of spatial data in geographic databases and is used in a wide variety of GIS applications. A topomap consists of multiple layers of data, where each layer depicts a specific characteristic of earth's surface such as vegetation cover, hydrology, topography etc. Colour is primarily used to differentiate between these layers. A sample map is shown in Figure 3.1. In this map, water bodies are shown in blue, grid lines in red, toponyms in black, contour lines in brown. For applications in GIS, each of these raster layers should be separated so that in each cell of a particular layer, there is one and only one thematic attribute or value. Automatic separation of different layers of information in topomaps poses an immense challenge due to the heavy interconnectedness of these layers. Our focus is on the generation of layer pertaining to contours, which has a tremendous application to geo scientific modelling due to the potential 3D representation of the terrain (which is a demanding tool from Geoscientists to take the advantage of visualization tools).

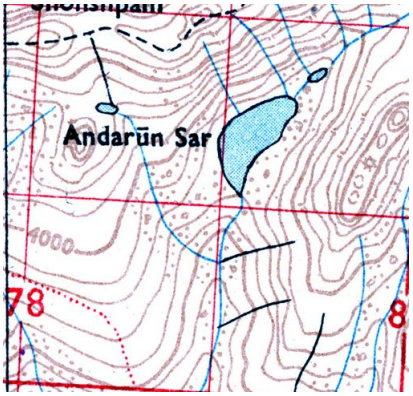


Figure 3.1: Sample Topomap

Contour lines are represented by curves with 2-4 pixels width when scanned at a resolution of 300dpi, and are brown in colour. The lines are labelled indicating the elevation values. Khotanzad et al. (KZ03) presented various challenges of automatic extraction and vectorization of contour lines from colour topomaps. A layer corresponding to contours will invariably have noise, discontinuities and gaps. The extraction of a contour layer thus introduces a challenging problem of filtering unwanted text, identification of gaps and filling them, for realistic automatic vectorization. This chapter focusses on separating the contour layer from topomap using colour clustering algorithm and then filtering text and noise from the extracted contour layer.

3.1 Extraction of Layers from Topographic Map

The first step in the conversion of topomaps is colour based separation of features. However existence of colour in topomaps introduces other dimensions to the problem : false colours, aliasing which are induced by the scanning process (KZ03).

Literature survey indicates that extensive work has been done in improving the segmentation of topomaps using various segmentation algorithms which are threshold based, histogram based, region based (region growing and merging), and hybrid approaches. Each of the segmentation algorithms use any one of the colour spaces like RGB, HIS, CIE Lab etc. Early segmentation methods

on topomaps concentrated on colour space selection. Ansoult et al. (ASL90) use the mean and variance of the hue channel for discriminating soil types on a digitized soil map. Ebi et al. (ELA94) transform the input RGB colour space into another colour space considering the chromaticity. Khotanzad et al. (KZ03) addressed the problem by proposing a method to build a colour key set for linear and area features of standard USGS topomaps and a non parametric clustering algorithm based on multidimensional histogram of the topomap. Yang Chen et al. (CWQ06b) extended this work beyond the standard USGS maps to common conditioned maps by using the features extracted from the gray version of the same colour map along with the colour coded map. Carol Rids et al. (RRA04) used colour edge detection - vector angle edge detector, with a saturation-based combination of hue and intensity planes, and colour clustering (minimum variance quantization) to extract the contour lines from colour coded maps.

Khotanzad et al. (KZ03) developed a colour key set for USGS topographic maps to address the problems associated with thin, closely spaced linear features and intersecting features. However, besides USGS topographic maps, other topographic maps in use, especially published in old age, are often non-USGS with poor conditions. Hence the focus of work has been to select the colour feature set, which can efficiently segment any kind of topographic map, than to improve the segmentation algorithm. Major steps of our algorithm are as follows:

1. Select the colour features of topomap using Principal Component Analysis.
2. Quantize the feature set using Uniform Quantization
3. Cluster the quantized feature set by a Histogram based Non Parametric Clustering Algorithm. (KB90)
4. Merge the clusters in order to obtain the layers of topomap by Spatial Analysis.

Each of the steps are now explained in detail in the following sections.

3.1.1 Feature Selection

Colour is perceived by humans as a combination of tristimuli R (red), G (green), and B (blue) which are usually called three primary colours. From R, G, B representation, we can derive other kinds of colour representations (spaces) by using either linear or non-linear transformations. Several colour spaces, such as $RGB, HSI, CIE\ L^*a^*b^*$ etc. are utilized in colour image segmentation. None of the colour spaces is better than the other towards representing all kinds of colour images. Selecting the best colour space is still one of the difficulties in colour image segmentation (CJSW01). To alleviate this problem in our work we propose a Hybrid colour space, with the best features for describing a particular topomap. We address the issue as a feature selection problem, thereby selecting best features from all the colour spaces for any given image. Feature selection is done using Principal Component Analysis (PCA).

The topomap is scanned and stored as a colour image using RGB colour space. Let this be denoted as I_{inp}. The colour features of various colour spaces like HSV, YC_bC_r, YIQ etc. are computed for each pixel of the image by transforming the RGB values as described below:

Colour Space Conversion

1. YC_bC_r is the colour space which separates luminance from the colour information. The luminance is encoded in the Y and the blueness and redness encoded in C_bC_r.

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.2999 & 0.587 & 0.114 \\ -0.16874 & -0.33126 & -0.5 \\ 0.5 & -0.41869 & -0.08131 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

   Let YI_{inp} represent the YC_bC_r transformed image.

2. YIQ is a matrix that contains the NTSC luminance (Y) and chrominance (I and Q) colour components as columns that are equivalent to the colours

in the RGB image. It is obtained from RGB by a linear transformation:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.2999 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

where $0 \leq R \leq 1, 0 \leq G \leq 1, 0 \leq B \leq 1$.

Let the YIQ transformed image be stored in $Y'I_{inp}$

3. The HSV (Hue-Saturation-Value) system is another commonly used colour space in image processing, which is more intuitive to human vision. Colour information is represented by hue and saturation values, converted from RGB using the following equations:

$$\begin{aligned} M &= max(R, G, B) \\ m &= min(R, G, B) \\ C &= M - m \\ H &= \arctan\left(\frac{\sqrt{3}(G-B)}{(R-G)+(R-B)}\right) \\ V &= M \\ S &= \begin{cases} 0 & \text{if } C = 0 \\ \frac{C}{V} & \text{otherwise} \end{cases} \end{aligned}$$

Let the HSV transformed image be stored in HI_{inp}.

4. CIE (Commission International de l'Eclairage) colour system was developed to represent perceptual uniformity, and thus meets the psychophysical need for a human observer. It has three primaries denoted as X,Y and Z. Any colour can be specified by the combination of X,Y and Z. The values of X,Y, and Z can be computed by a linear transformation from RGB tristimulus coordinates. In particular, the transformation matrix for the NTSC (National Television System Commission, United States) receiver primary

system is

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.607 & 0.174 & 0.200 \\ 0.299 & 0.587 & 0.114 \\ 0.000 & 0.066 & 1.116 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Let the XYZ transformed image be stored in CI_{inp}.

5. A number of CIE spaces can be created once the XYZ tristimulus coordinates are known. CIE (L*a*b*) space is one typical example. It can be obtained through nonlinear transformations of X, Y, and Z values.

$$L^* = 116 \left(\sqrt[3]{\frac{Y}{Y_0}} \right) - 16$$

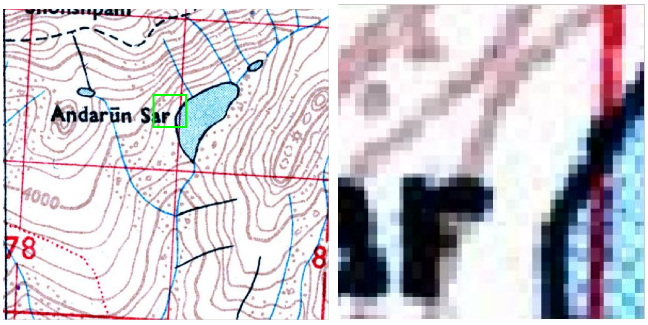
$$a^* = 500 \left[\sqrt[3]{\frac{X}{X_0}} - \sqrt[3]{\frac{Y}{Y_0}} \right]$$

$$b^* = 200 \left[\sqrt[3]{\frac{Y}{Y_0}} - \sqrt[3]{\frac{Z}{Z_0}} \right]$$

where $\frac{X}{X_0} > 0.01, \frac{Y}{Y_0} > 0.01, \frac{Z}{Z_0} > 0.01$. (X_0, Y_0, Z_0) are X, Y, Z values for the standard white.

Let the transformed image be stored in LI_{inp}.

A data set is generated from all the colour transformed images as described in the Procedure 1. Figure 3.2 shows respectively a cropped image shown in a green rectangle and some of its colour features. Algorithm 3.1 describes the steps to derive colour features that are used for further processing.



(a) Input Topomap (b) Cropped Image

R	G	B	H	S	V	Y	I	Q	Y	Cb	Cr	X	Y	Z	L*	a*	b*
207	167	176	0.9625	0.193237	207	179.9836	20.94382	11.26261	154.639	-1.47404	17.42774	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
255	226	231	0.971264	0.113725	255	235.2392	15.67464	7.69049	202.0933	-1.60044	12.88208	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
255	250	250	0	0.019608	255	251.4947	2.979729	1.057487	216.0526	-0.23916	2.698039	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
255	255	251	0.166667	0.015686	255	254.5439	1.286228	-1.24565	218.6711	-1.2549	0.787671	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
249	255	253	0.444444	0.023529	255	252.9783	-2.93256	-1.89181	217.3262	0.512871	-1.99048	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
245	255	253	0.466667	0.039216	255	251.7826	-5.31634	-2.7378	216.2991	1.105765	-3.74734	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
243	255	251	0.444444	0.047059	255	250.9567	-5.86512	-3.78362	215.5897	0.52378	-4.48292	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
244	255	250	0.424242	0.043137	255	251.1416	-4.94762	-3.88354	215.7485	-0.06366	-3.97227	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
246	255	250	0.407407	0.035294	255	251.7395	-3.75573	-3.46054	216.2621	-0.36011	-3.09384	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
248	254	250	0.388889	0.023622	254	251.7503	-2.28945	-2.51464	216.2716	-0.36556	-1.84762	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
252	255	253	0.388889	0.011765	255	253.8752	-1.14472	-1.25732	218.0966	0.0682	-0.67283	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
255	255	253	0.166667	0.007843	255	254.772	0.643114	-0.62283	218.8669	-0.37647	0.644816	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
255	254	253	0.083333	0.007843	255	254.1849	0.917503	-0.09992	218.3628	-0.08548	1.012604	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
255	251	253	0.916667	0.015686	255	252.4238	1.740669	1.468816	216.8504	0.787498	2.115969	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
249	230	236	0.947368	0.076305	249	236.3639	9.393626	5.88693	203.0586	0.321008	8.418494	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
218	191	200	0.944444	0.123853	218	200.0975	13.19652	8.513148	171.9125	0.452867	11.71794	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
203	170	181	0.944444	0.162562	203	181.1191	16.12908	10.40496	155.6137	0.441957	14.21038	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
205	169	181	0.944444	0.17561	205	181.1299	17.59536	11.35086	155.6232	0.436502	15.4566	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
188	150	163	0.942982	0.202128	188	162.8418	18.4657	12.08527	139.917	0.579271	16.2636	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
187	154	165	0.944444	0.176471	187	165.1191	16.12908	10.40496	141.8725	0.441957	14.21038	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
192	162	170	0.955556	0.15625	192	171.8802	15.30592	8.836227	147.6791	-0.43102	13.10701	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
204	183	188	0.960317	0.102941	204	189.8478	10.90708	5.998511	163.1095	-0.41465	9.368353	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
229	215	215	0	0.061135	229	219.1851	8.34324	2.960963	188.3048	-1.57317	6.65098	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
249	244	241	0.0625	0.032129	249	245.1526	3.9444	0.123247	210.6059	-1.5568	2.912322	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
255	254	249	0.138889	0.023529	255	253.7288	2.203731	-1.34557	217.9712	-1.84234	1.298314	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
248	252	253	0.533333	0.019763	253	250.9183	-2.70534	-0.53458	215.557	1.534071	-1.32633	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
248	255	255	0.5	0.027451	255	252.9074	-4.17162	-1.48048	217.2652	1.539525	-2.57255	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806
163	171	192	0.62069	0.151042	192	171.003	-11.5203	4.847702	146.9233	10.91128	-4.51174	0.964279	0.999969	0.825089	99.99882	0.018274	-0.01806

(c) Colour Features of the Cropped Image

Figure 3.2: Feature Set Generation

Procedure 1 FEATGEN(I_{inp})

// Computes the colour features of input image //

Input:

I_{inp}: RGB topomap image, 3D matrix of size $[N_r,\ N_c,\ 3]$

Output:

X: Feature set, 2D matrix of size $[N_r \times N_c,\ 18]$

Var:

F: 3D matrix of size $[N_r,\ N_c,\ 18]$

$i,\ j,\ k$: Loop indices

Method:

1: **for** $i \leftarrow 1$ *to* N_r **do**
2: **for** $j \leftarrow 1$ *to* N_c **do**
3: $F(i,j,1) \leftarrow I_{inp}(i,j,1)$ //R//
4: $F(i,j,2) \leftarrow I_{inp}(i,j,2)$ //G//
5: $F(i,j,3) \leftarrow I_{inp}(i,j,3)$ //B//
6: $F(i,j,4) \leftarrow HI_{inp}(i,j,1)$ // H: From Section 3.1.1, Step 3//
7: $F(i,j,5) \leftarrow HI_{inp}(i,j,2)$ // S: From Section 3.1.1, Step 3//
8: $F(i,j,6) \leftarrow HI_{inp}(i,j,3)$ //V: From Section 3.1.1, Step 3//
9: $F(i,j,7) \leftarrow YI_{inp}(i,j,1)$ // Y: From Section 3.1.1, Step 1//
10: $F(i,j,8) \leftarrow YI_{inp}(i,j,2)$ //C_b: From Section 3.1.1, Step 1//
11: $F(i,j,9) \leftarrow YI_{inp}(i,j,3)$ // C_r: From Section 3.1.1, Step 1//
12: $F(i,j,10) \leftarrow Y^1I_{inp}(i,j,1)$ // Y: From Section 3.1.1, Step 2//
13: $F(i,j,11) \leftarrow Y^1I_{inp}(i,j,2)$ // I: From Section 3.1.1, Step 2//
14: $F(i,j,12) \leftarrow Y^1I_{inp}(i,j,3)$ //Q: From Section 3.1.1, Step 2//
15: $F(i,j,13) \leftarrow CI_{inp}(i,j,1)$ //X: From Section 3.1.1, Step 4//
16: $F(i,j,14) \leftarrow CI_{inp}(i,j,2)$ //Y: From Section 3.1.1, Step 4//
17: $F(i,j,15) \leftarrow CI_{inp}(i,j,3)$ //Z: From Section 3.1.1, Step 4//
18: $F(i,j,16) \leftarrow LI_{inp}(i,j,1)$ //L*: From Section 3.1.1, Step 5//
19: $F(i,j,17) \leftarrow LI_{inp}(i,j,2)$ //a*: From Section 3.1.1, Step 5//
20: $F(i,j,18) \leftarrow LI_{inp}(i,j,3)$ //b*: From Section 3.1.1, Step 5//
21: **end for**
22: **end for**
 //F is reshaped to a 2-D matrix, X //
23: **for** $i \leftarrow 1$ *to* N_r **do**
24: **for** $j \leftarrow 1$ *to* N_c **do**
25: **for** $k \leftarrow 1$ *to* 18 **do**
26: $X((i-1) \times N_c + j, k) \leftarrow F(i,j,k)$
27: **end for**
28: **end for**
29: **end for**
30: **return** X

end FEATGEN

3.1.2 Quantization

The colour of pixel is represented by three dimensional vector, RGB where each channel can have a value between 0 and 255. A 24-bit colour image contains at most $2^{24} = 16777216$ different colours. Though a limited set of colours are used in topomap for representing the various features, scanning process intro-

Algorithm 3.1 FEATSEL(I_{inp})

//Feature Selection using PCA //
Input:
I_{inp}: RGB topomap image of size $[N_r,\ N_c,\ 3]$
Output:
I_f: PCA transformed image
Var:
X: Array of size $[N_r \times N_c,\ 18]$
Y: Array of size $[N_r \times N_c,\ 3]$
$i,\ j,\ k$: Loop indices
Method:

1: $X \leftarrow FEATGEN(I_{inp})$
2: Transform X along 3 Principal Components to obtain Y
//Convert Y to a 3D matrix of size $[N_r,\ N_c,\ 3]$//
3: **for** $i \leftarrow 1$ *to* N_r **do**
4: **for** $j \leftarrow 1$ *to* N_c **do**
5: **for** $k \leftarrow 1$ *to* 3 **do**
6: $I_f(i,j,k) = Y((i-1) \times N_c + j, k)$
7: **end for**
8: **end for**
9: **end for**
10: **return** I_f
end FEATSEL

Computational Complexity of FEATSEL	
Step 1	$N_r \times N_c \times 18$
Step 2	$O(N^3)$
Steps 3-9	$N_r \times N_c \times 3$
Algorithm Complexity: $O(N^3)$ where N is $Max(N_r, N_c)$	

duces false colours. For example, in a simple scanned USGS map consisting only of blue, brown, black and purple, over 42000 distinct colours can be detected (PT08). Thus by employing colour quantization full colour RGB images, where each pixel is typically described by three 8-bit colour samples, is represented in an approximate fashion by a relatively small number of colours. Thus the full colour space, consisting of about 16 million pixels (2^{24}), is divided into a small number of regions, and for each region, a single representative colour is used for each pixel that falls into that region.

The colour topomap which is represented using Hybrid colour space is quantized as described in the Algorithm 3.2. Figure 3.3 shows the portion of quantized data for the cropped image shown in Figure 3.2.

R	G	B	H	S	V	Y	I	Q	Y	Cb	Cr	X	Y	Z	L*	a*	b*
14	13	13	5	5	14	13	6	6	12	5	6	10	5	5	5	5	5
15	15	15	5	5	15	15	6	6	14	5	6	10	5	5	5	5	5
15	15	15	5	5	15	15	6	5	15	5	6	10	5	5	5	5	5
15	15	15	5	5	15	15	5	5	15	5	5	10	5	5	5	5	5
15	15	15	5	5	15	15	5	5	15	5	5	10	5	5	5	5	5
15	15	15	5	5	15	15	5	5	15	5	5	10	5	5	5	5	5
15	15	15	5	5	15	15	5	5	15	5	5	10	5	5	5	5	5
15	15	15	5	5	15	15	5	5	15	5	5	10	5	5	5	5	5
15	15	15	5	5	15	15	5	5	15	5	5	10	5	5	5	5	5
15	15	15	5	5	15	15	5	5	15	5	5	10	5	5	5	5	5
15	15	15	5	5	15	15	5	5	15	5	5	10	5	5	5	5	5
15	15	15	5	5	15	15	5	5	15	5	5	10	5	5	5	5	5
15	15	15	5	5	15	15	5	5	15	5	5	10	5	5	5	5	5
15	15	15	5	5	15	15	5	5	15	5	5	10	5	5	5	5	5
15	15	15	5	5	15	15	6	6	14	5	6	10	5	5	5	5	5
15	14	14	5	5	15	14	6	6	13	5	6	10	5	5	5	5	5
14	13	13	5	5	14	13	6	6	12	5	6	10	5	5	5	5	5
14	13	13	5	5	14	13	6	6	12	5	6	10	5	5	5	5	5
14	12	12	5	5	14	12	6	6	11	5	6	10	5	5	5	5	5
14	12	13	5	5	14	13	6	6	12	5	6	10	5	5	5	5	5
14	12	13	5	5	14	13	6	6	12	5	6	10	5	5	5	5	5
14	13	14	5	5	14	14	6	6	12	5	6	10	5	5	5	5	5
15	15	15	5	5	15	15	6	6	14	5	6	10	5	5	5	5	5
15	15	15	5	5	15	15	6	5	15	5	6	10	5	5	5	5	5
15	15	15	5	5	15	15	5	5	15	5	5	10	5	5	5	5	5
15	15	15	5	5	15	15	5	5	15	5	5	10	5	5	5	5	5
15	15	15	5	5	15	15	5	5	15	5	5	10	5	5	5	5	5
12	13	14	5	5	14	13	5	6	12	6	5	10	5	5	5	5	5

Figure 3.3: Quantized Features of the Cropped Image shown in Figure 3.2

3.1.3 Clustering Algorithm

The topomap segmentation has been performed using a non parametric histogram based clustering algorithm proposed by Khotanzad et al. (KB90). The quantized image is clustered by computing the histogram and finding the peaks of histogram using peak climbing approach. This algorithm is non parametric i.e. requires

Algorithm 3.2 IMGQUANT(I_f)

// Quantize the Image //
Input:
I_f: PCA transformed colour image of size $[N_r,\ N_c,\ 3]$
(From Algorithm 3.1)
Output:
I_q: Quantized Image of size $[N_r,\ N_c\ 3]$
Var:
Q_C: Quantization levels, Set of 3 values
i, j, k: Loop indices
Method:

1: PCA transformed colour image, I_f is represented as a cube with its three components P^1, P^2, P^3 along Cartesian axis X,Y,Z respectively.
2: The minimum and maximum values along each axis of the cube are chosen from the input data i.e for example $X_{min} = P^1_{min}, X_{max} = P^1_{max}$
3: The cube is partitioned into sets of non overlapping cells. If the quantization levels are $(Q_C, C = P^1, P^2, P^3)$, the whole cube is divided into $Q_{P^1} \times Q_{P^2} \times Q_{P^3}$ cells.
//The tristimuli values of each pixel of the input image are mapped to a quantization level between 1 & Q_C//
4: **for** $i \leftarrow 1$ *to* 3 **do**
5: $incr \leftarrow (P^i_{max} - P^i_{min})/Q_{P^i}$
6: **for** $j \leftarrow 1$ *to* N_r **do**
7: **for** $k \leftarrow 1$ *to* N_c **do**
8: $I_q(j,k,i) = \lfloor \frac{I_f(j,k,i) - P^i_{min}}{incr} \rfloor + 1$
9: **end for**
10: **end for**
11: **end for**
12: **return** I_q
end IMGQUANT

Computational Complexity of IMGQUANT*	
Step 2	$N_r \times N_c \times 3$
Steps 4,6,7	$N_r \times N_c \times 3$
Algorithm Complexity: $O(N^2)$ where N is $Max(N_r, N_c)$	

*Steps with constant time complexity are not mentioned

no parameters from the user as input. A three dimensional histogram of the quantized image is computed by counting the number of pixels in each cell of the cube. Clustering is performed by locating the peaks of the multidimensional histogram using peak climbing algorithm as described in the Algorithm 3.3.

It can be noted that the method is completely automatic and can be applied on any colour image. It is amenable to parallel processing as the image can be divided and method applied on different parts of the image.

The results of applying Algorithms 3.2, 3.3 on the Image shown in Figure 3.1 are shown in the Figure 3.4. 13 clusters are obtained from topomap 1. It can be noted that number of clusters obtained is greater than the number of distinct colours present in a topographic map, due to presence of false colours and aliasing. Hence a post processing stage is necessary which can map the clusters to the layers of topomap. However, as the PCA transforms the colour image, once the clusters are obtained from initial clustering, the same space cannot be used for further processing of clusters. For example, distance metrics cannot be used to merge the clusters. Hence, we propose a method which uses spatial properties of pixels to merge the clusters as described in section 3.1.4.

3.1.4 Merging the Clusters

The clusters obtained on a PCA transformed colour topomap, are mapped to the layers. Prior to applying spatial analysis, clusters which belong to noise should be removed. In addition to noise, a cluster which belongs to background should also be removed as further processing is based on the neighbourhood analysis (number of pixels which belong to a cluster different from the centre pixel). The steps are shown in Procedure 2. $T1$ is the threshold on ratio of size of cluster to total image size, used to remove clusters belonging to noise.

Table 3.1 shows the size and ratio of size to total image size, for each of the clusters shown in Figure 3.1. The clusters 4, 7, 8, 9, 10 belong to noise and cluster 13 belongs to background. Hence only clusters which belong to the thematic layers of topomap like contours, vegetation, water bodies etc. are considered for further analysis by removing noise, background as described in the Algorithm 3.4.

Algorithm 3.3 CLUSTER(I_q)

// Clustering algorithm (KB90): //

Input:

I_q: Quantized image of size $[N_r,\ N_c,\ 3]$

Output:

C, $\{C_t, t = 1\ to\ N_{tc}\}$: Set of Clusters C_t where N_{tc} is Number of clusters

Method:

1: A three dimensional histogram of the quantized image, I_q is computed.

2: The clustering operation requires the examination of all the neighbouring cells of all non-empty cells. The number of neighbours for each discrete cell is a function of the dimension of the space. The number of possible neighbours in the three dimensional representation is 26.

3: By examining the counts of neighbours of a particular cell, a link is established between that cell and the closest cell having the largest count in the neighbourhood termed as parent. If two or more neighbouring cells have the same count for a particular cell, one of them is arbitrarily chosen as a parent. //Method assumes that two or more neighbouring peaks belong to the same cluster//

4: At the end of the link assignment each cell is linked to one parent cell, but can be parent of more than one cell.

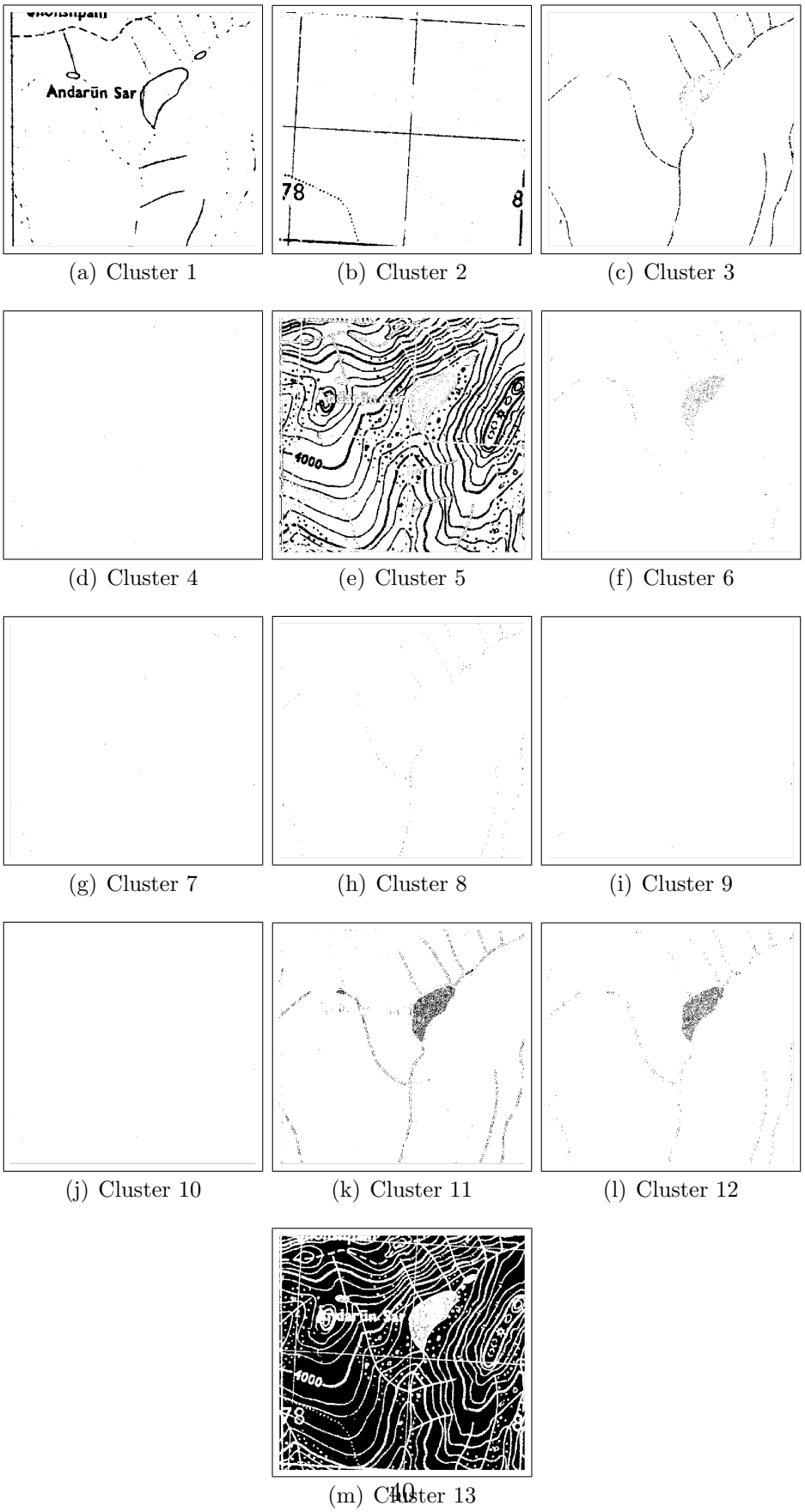
5: A peak is defined as being a cell with the largest density in the neighbourhood, i.e. a cell with no parent.

6: A peak and all the cells which are linked to it are taken as a distinct cluster representing a mode of the histogram.

7: The clustering process starts from each peak and identifies all the other cells linked to its peak by following the connectivity of the cells through the links. Clusters C_t, which are equal to the number of peaks of histogram are formed.

8: **return** C

end CLUSTER



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3

(d) Cluster 4 (e) Cluster 5 (f) Cluster 6

(g) Cluster 7 (h) Cluster 8 (i) Cluster 9

(j) Cluster 10 (k) Cluster 11 (l) Cluster 12

(m) Cluster 13

Figure 3.4: Clusters obtained from Topomap in Figure 3.1

Procedure 2: REMOVECLUS(C)

// Remove noise and background clusters //

Input:

C, $\{C_t, t = 1\ to\ N_{tc}\}$: Set of Clusters obtained from Algorithm 3.3

Output:

RC, $\{RC_t, t = 1\ to\ N_{rc}\}$: Set of Clusters obtained after removing noise, background

Var:

S^z: List of size N_{tc}

T_1: 0.0001 (Threshold to remove noise)

i: Loop index

S^z_M: Maximum cluster size

Method:

1: Number of pixels belonging to each cluster i.e. cluster size is computed for each cluster. Let this be stored in list S^z.
2: $N_{rc} \leftarrow 0$
3: $S^z_M \leftarrow Max(S^z)$ //Function to find Maximum of numbers//
4: **for** $i \leftarrow 1\ to\ N_{tc}$ **do**
5: **if** $S^z(i) = S^z_M$ **then**
6: continue
//The cluster of maximum size, belongs to background//
7: **end if**
//Clusters which have relatively very less size compared to the image size, are considered as noise//
8: **if** $\frac{S^z_i}{Sum(S^z)} > T_1$ **then**
9: $N_{rc} \leftarrow N_{rc} + 1$
10: $RC_{N_{rc}} \leftarrow C_i$
11: **end if**
12: **end for**
13: **return** RC

end REMOVECLUS

Computational Complexity of MERGECLUS*	
Step 1	N_{tc} (small integer)
Step 2, 6	N_{rc} (small integer)
Steps 3	arbitrary (small integer)
Step 4	$N_r \times N_c$
Step 18	$O(N^2_{rc})$
Algorithm Complexity: $O(N^2)$ N is $Max(N_r, N_c)$	

*Steps with constant time complexity are not mentioned

Algorithm 3.4 MERGECLUS(C)

// Spatial analysis of clusters //

Input:

C, $\{C_t, t = 1\ to\ N_{tc}\}$: Set of Clusters obtained from Algorithm 3.3

Output:

L, $\{L_t, t = 1\ to\ N_l\}$: Set of merged Layers of topomap where

N_l: Number of layers

Var:

RC, $\{RC_t, t = 1\ to\ N_{rc}\}$: Set of Clusters obtained after removing noise, background

n_{ij}^p: Integer

cnt: 2D Array of size $[N_{rc},\ N_{rc}]$ initialized to 0

A_{rc}: Matrix of size $[N_{rc}, N_{rc}]$

T_2: Threshold (Arrived based on the cluster size)

w: Size of window

$i,\ j$: Loop indices

Method:

1: $RC \leftarrow$ REMOVECLUS(C) //Removes noise and background clusters//
2: **for** $i \leftarrow 1\ to\ N_{rc} - 1$ **do**
3: **while** size of RC_i does not change between the two iterations **do**
4: **for** each object pixel p in the cluster, RC_i **do**
5: Consider a $w\ \times\ w$ neighbourhood of p.
6: **for** $j \leftarrow i + 1\ to\ N_{rc}$ **do**
7: Count the number of pixels in the neighbourhood which belong to RC_j. Let this be n_{ij}^p
8: **if** $n_{ij}^p > (w^2 - 1)/2$ **then**
9: Assign pixel p to cluster RC_j
10: $cnt(i, j) \leftarrow cnt(i, j) + 1$
 //Since more than half pixels belong to RC_j, further clusters need not be checked as the condition will be false //
11: break
12: **end if**
13: **end for**
14: **end for**
15: **end while**
16: **end for**
17: An adjacency matrix, A_{rc} of size $N_{rc},\ N_{rc}$ is formed from the matrix cnt. $A(i, j)$ is assigned with 1 when $cnt(i, j)$ is maximum in the column j and greater than a threshold T_2, otherwise 0
18: Components are obtained using backtracking algorithm on A
19: Clusters of each component are merged to obtain one layer of topomap. Number of layers equal to the number of components (N_l), L_t are obtained
20: **return** L

end MERGECLUS

Table 3.1: Cluster properties shown in Figure 3.1

Cluster	Size	Size/Image Size	Layer
1	9227	0.0377	Layer 1
2	7016	0.0286	Layer 2
3	3185	0.0130	Layer 3
4	9	0.00003	Noise
5	54785	0.2237	Layer 4
6	613	0.0025	Layer 3
7	12	0.00004	Noise
8	225	0.0009	Noise
9	9	0.00003	Noise
10	5	0.00002	Noise
11	4415	0.0180	Layer 3
12	2120	0.0087	Layer 3
13	163283	0.6667	Background

Table 3.2: Size of clusters after every iteration

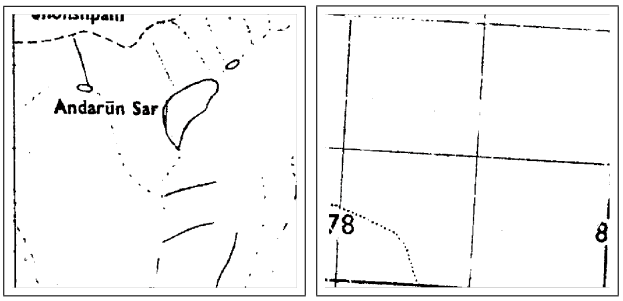
	1	2	3	5	6	11	12
0	9227	7016	3185	54785	613	4415	2120
1	9050	7044	3239	54874	613	4421	2120
2	9050	6980	3239	54938	613	4421	2120
3	9050	6980	3109	54942	613	4547	2120
4	9050	6980	3109	54942	613	4547	2120
5	9050	6980	3109	54942	315	4826	2139
6	9050	6980	3109	54942	315	4548	2417

Count Matrix

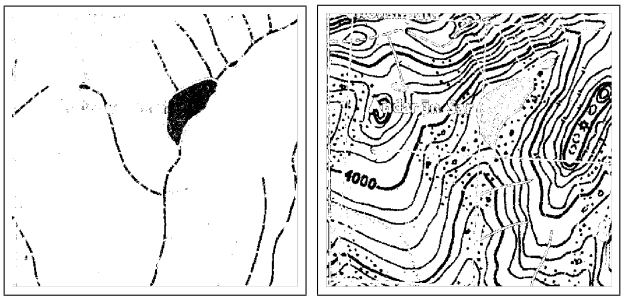
	1	2	3	5	6	11	12
1	0	28	54	89	0	6	0
2	0	0	0	64	0	0	0
3	0	0	0	4	0	135	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	279	19
11	0	0	0	0	0	0	278
12	0	0	0	0	0	0	0

Adjacency Matrix

	1	2	3	5	6	11	12
1	1	0	0	0	0	0	0
2	0	1	0	0	0	0	0
3	0	0	0	0	0	1	0
5	0	0	0	1	0	0	0
6	0	0	0	0	0	1	0
11	0	0	1	0	1	0	1
12	0	0	0	0	0	1	0



(a) Layer 1 (b) Layer 2



(c) Layer 3 (d) Layer 4

Figure 3.5: Merged Layers of Topomap in Figure 3.1

After removing noise and background, 7 clusters, 1, 2, 3, 5, 6, 11, 12 are obtained, which are further processed . For the 7 object clusters, Table 3.2 shows the change in size after every iteration (of outer for loop in Algorithm 3.4). Matrix *cnt*, which stores the number of pixels assigned from one cluster to other clusters is shown. Adjacency matrix formed from *cnt*, using threshold T_2 is also shown. It can be observed that clusters 1 and 5 are not merged though some pixels (89) are assigned from cluster 1 to cluster 5 as the number is less than threshold $T2$. From the adjacency matrix, 4 components are obtained. Hence cluster 1 is mapped to Layer 1, cluster 2 to layer 2, clusters 3, 6, 11, 12 are merged to obtain layer 3 and cluster 5 to layer 4 as shown in the last column of the Table 3.1. The Figure 3.5 shows the merged clusters, which form the layers of topomap shown in Figure 3.1.

3.1.5 Results

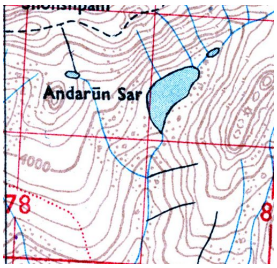
The significance of using a feature selection on all the colour features for clustering the topomap can be observed by comparing the same clustering algorithm applied against a RGB colour space. For example for the topomaps shown in Figures 3.6(a), 3.6(e) it is observed that clustering done with PCA transformed image gives more number of contour pixels compared to RGB space which have been shown in Figure 3.6(d), 3.6(h) . It is visually very apparent that PCA space performs better than RGB space. However to quantify the performance, the accuracy of segmentation has been computed using the method proposed in (LB10).

Accuracy Assessment of Contour Layer Extraction

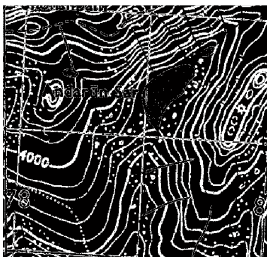
The results are illustrated using the test images shown in Figure 3.7. The test images are parts of topomaps prepared by Survey Of India, of scale 1:50000. They are scanned with a resolution of 300dpi. The first step is systematic sampling over the set of processed map documents, i.e training data is prepared for each sample image, by manually assigning pixels to the layer of topomap it belongs to. For each image a limited random number of pixels have been selected under each layer. Care has been taken to select the pixels all across the topomap. After employing clustering, the layer index of each of the sampled pixel is stored. A confusion matrix is created using the clustered and labelled pixels, as shown in 3.3. Measures derived from the confusion matrix are *Recall* and *Precision* for the contour layer.

Table 3.3: Confusion Matrix

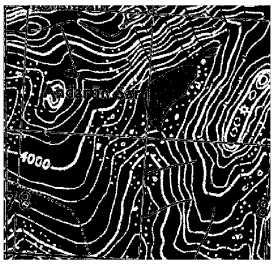
		Predicted class	
		Contour	Other
Test outcome	Contour	True Positive(tp)	False Positive(fp)
	Other	True Negative(tn)	False Negative(fn)



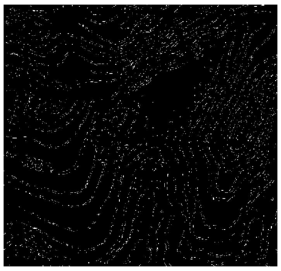
(a) Topomap 1



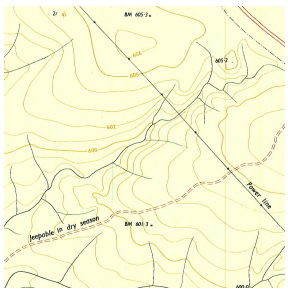
(b) Contour layer obtained using RGB



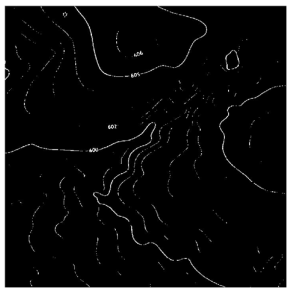
(c) Contour layer obtained using PCA



(d) Pixels in PCA and not in RGB

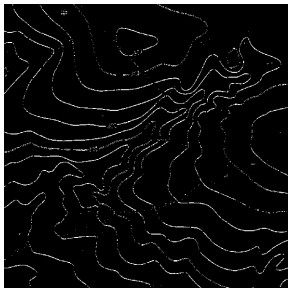


(e) Topomap 2



(f) Contour layer obtained using RGB

(g) Contour layer obtained using PCA



(h) Pixels in PCA and not in RGB

Figure 3.6: Comparison of Contour layer Extraction using RGB and PCA

Precision is a measure of the accuracy provided that contour pixels have been predicted. It is defined by:

$$Precision = tp/(tp + fp)$$

where tp and fp are the numbers of true positive and false positive predictions for the considered class.

Recall is a measure of the ability of a prediction model to select instances of contour pixels from a data set. It is commonly also called sensitivity, and corresponds to the true positive rate. It is defined by the formula:

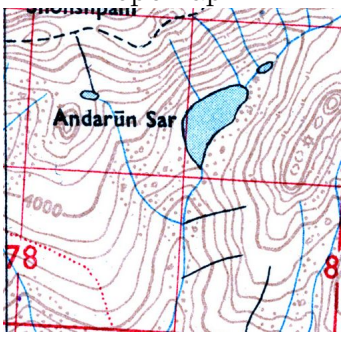
$$Recall = Sensitivity = tp/(tp + fn)$$

where tp and fn are the numbers of true positive and false negative predictions for the considered class. tp + fn is the total number of test examples of the considered class.

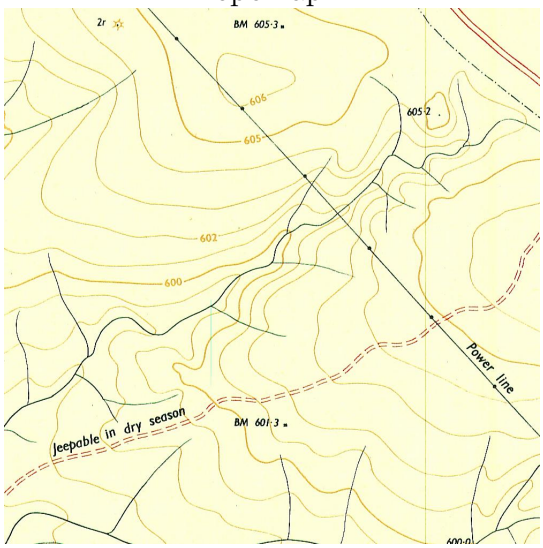
Table 3.4 shows the results on 5132 pixels sampled from twelve maps, shown in Figure 3.7. It can be seen that the same clustering algorithm performs better in the case of PCA space than the RGB space. For few images, using RGB space contour layer could not be separated from background. For images Topomap 8 and Topomap 10, using quantization level 16, along each axis, the contour layer could not be separated from background. However using quantization level 32, the contour layer could be obtained. Hence the algorithm is sensitive to the quantization level.

It can be noted that the proposed method can be used on any colour topographic document.

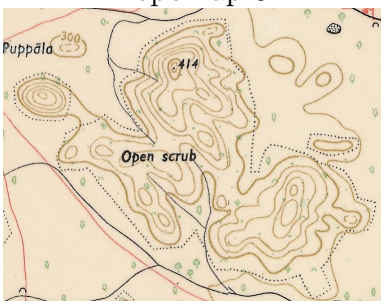
Topomap 1



Topomap 2



Topomap 3



Topomap 4

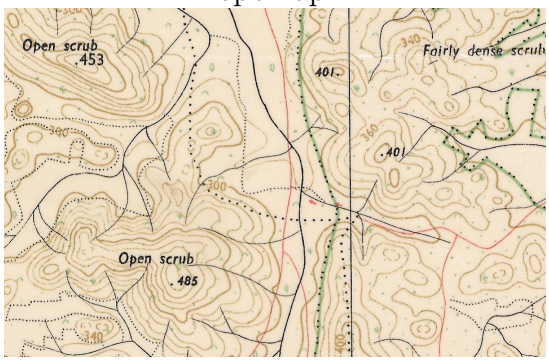
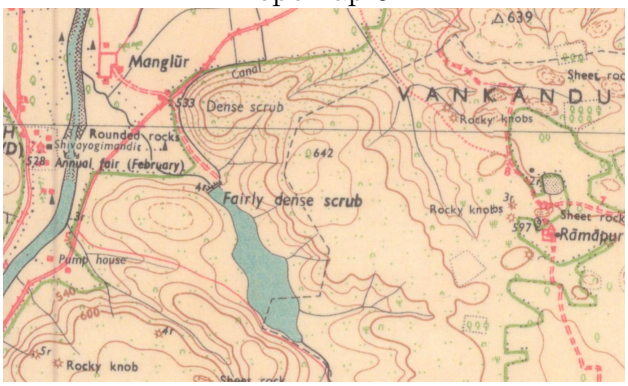
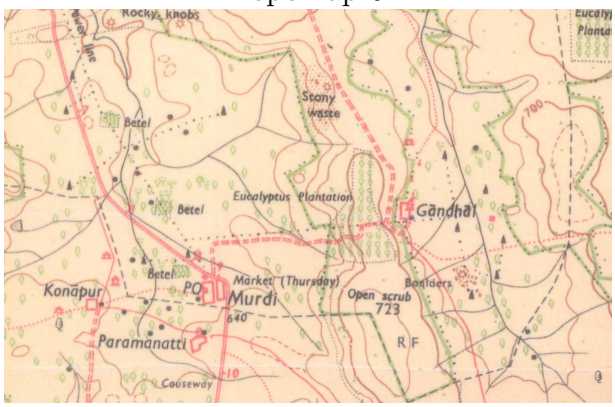


Figure 3.7: Test Images

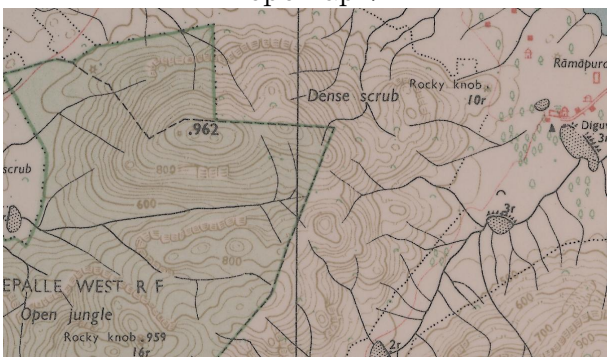
Topomap 5



Topomap 6



Topomap 7



Topomap 8

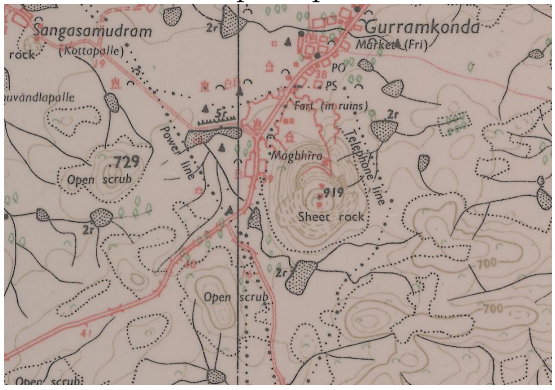
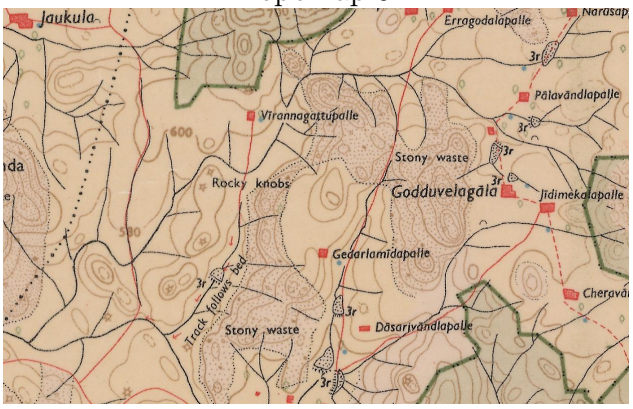
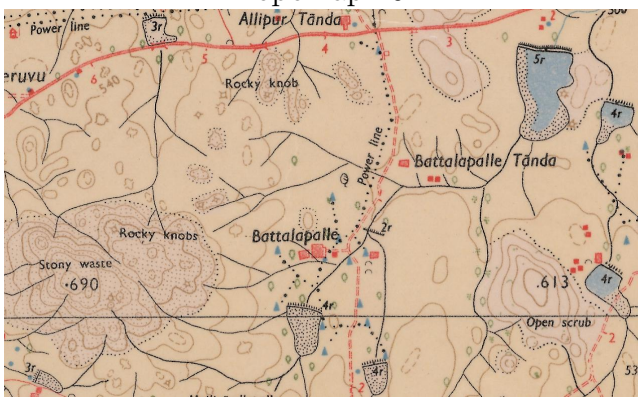


Figure 3.7: Test Images

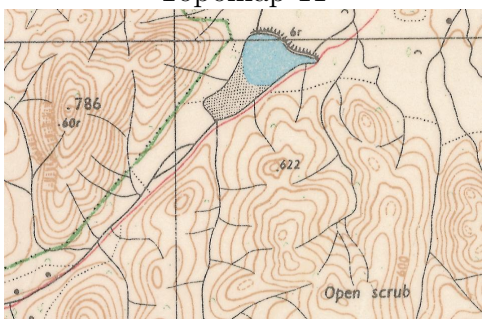
Topomap 9



Topomap 10



Topomap 11



Topomap 12

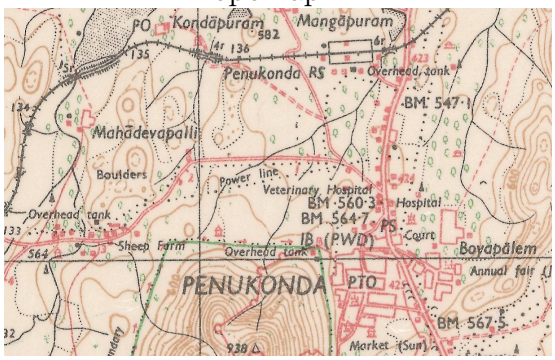
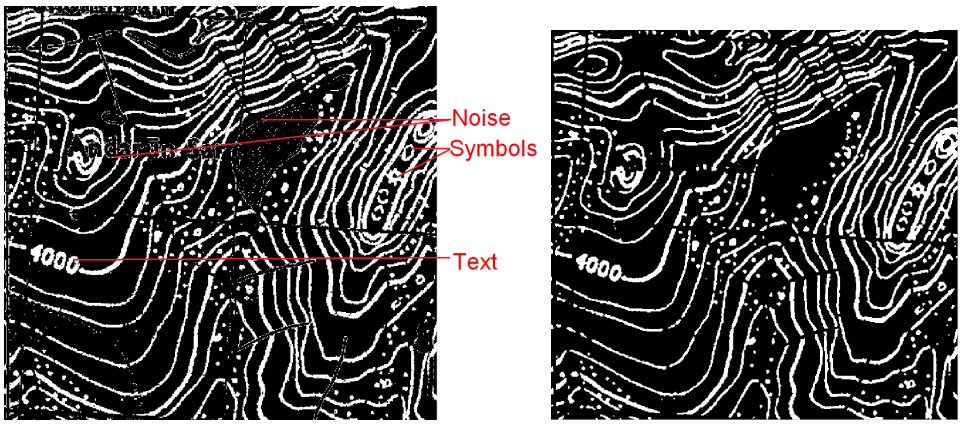


Figure 3.7: Test Images

Table 3.4: Results of Accuracy Assessment

Image	Size	Quant	No of pixels		RGB		PCA	
Map		level	Total	Contour	Precision	Recall	Precision	Recall
1	484 X 506	16	336	77	89.16	96.10	96.15	97.40
2	1001 X 1001	16	242	98	100	59.18	100	71.43
3	583 X 755	16	296	85	51.27	95.29	100	87.06
4	695 X 1083	16	400	131	NaN	0	98.77	61.07
5	746 X 1243	16	402	141	98.21	78.01	99.09	77.3
6	787 X 1212	16	414	124	97.98	86.29	99.07	86.29
7	695 X 1205	16	401	105	NaN	0	79.46	84.76
8	760 X 1092	32	560	169	NaN	0	100	47.83
9	790 X 1249	16	332	104	NaN	0	49.15	83.65
10	774 X 1264	32	487	100	NaN	0	97.37	74
11	617 X 954	16	672	197	62.89	92.89	99.44	89.85
12	693 X 1095	16	590	112	72.06	87.5	98.18	96.43



(a) Contour layer from clustering

(b) Contour layer after median filter

Figure 3.8: Median Filtering

3.2 Filtering of Contour Layer

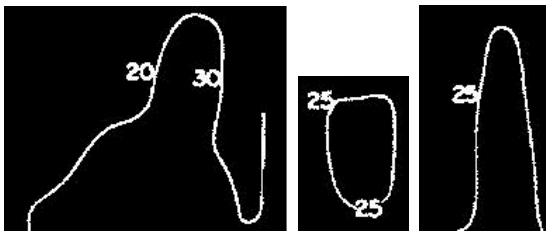
A topographic map is thematically layered based on the colour. We take contour layer for our further work. Figure 3.8(a) shows a contour layer extracted from a topomap (Ref Figure 3.5(d)) which is not clean due to presence of noise, text, symbols. This is due to the fact that segmentation is not efficient due to various problems induced by scanning and some due to inherent characteristics of topomap. Apart from the contour lines, which connect the points of equal elevation, the contour layer also has symbols and altitude tags. The text cannot be filtered in the Extraction stage as the altitude values are of the same colour as contour lines, and contour layer extraction is based on colour. Filtering stage consists of algorithms to filter noise and extract text to generate a clean contour image. Filtering the altitude tags from contour layer is an important pre cursor to contour layer vectorization.

The noise can be filtered using the standard algorithms like median filter. Figure 3.8(b) shows the contour layer obtained after applying median filter (Object is shown in white and background in black for clarity).

However text has to be handled in a different way. Features based on spatial arrangement and intensity distributions of pixels are necessary to separate text, which are of the same colour as lines from the contour layer. Depending on the

(a) (b) (c)

Figure 3.9: Text not Connected to Contour Lines



(a) (b) (c)

Figure 3.10: Text Connected to Contour Lines

position of altitude values, the tags can be divided into two parts

1. Text in between two contour lines, not connected to any of the contour lines, shown in Figure 3.9.

2. Text connected to a contour line(s), shown in Figure 3.10.

Automatic segmentation of text/symbols from documents has been one of the fundamental aims in graphics recognition. One way has been to separate text from documents based on colour - This is applicable when text is represented using a different colour from the other graphics in the document. The emphasis in this case is again on colour segmentation/clustering as described in previous section. For example Joachim Pouderoux et al. (PGPG07) extracted toponyms from map based on colour. Second branch of research is to extract text from documents without colour features. To extract text not connected to the graphics of document, Fletcher and Kasturi (FK88) used connected components and some

properties like Area etc. Morphology was used by Luo et al. (HGI95). Cao et al. (RC01) proposed a method to extract text touching to the graphics, but their method is done on a vectorized image. Tombre et al. (KSL$^+$02) proposed a method based on connected components. P.P.Roy et al. (PVL$^+$08) developed a system to segment text using colour, and further by using geometric features of connected components and skeleton information in each layer.

The proposed method filters text not connected to contour layers by a method proposed by Fletcher et al. Text connected to contours is filtered using spatial arrangement and intensity distributions. Major steps in the process are:

1. Perform Connected Component Analysis (CCA) and compute properties of each component like Area of minimum bounding rectangle (MBR), Aspect ratio of MBR, total object pixels etc.

2. Filter text/symbols not connected to contours

3. Filter text connected to contour lines

4. Filter noise

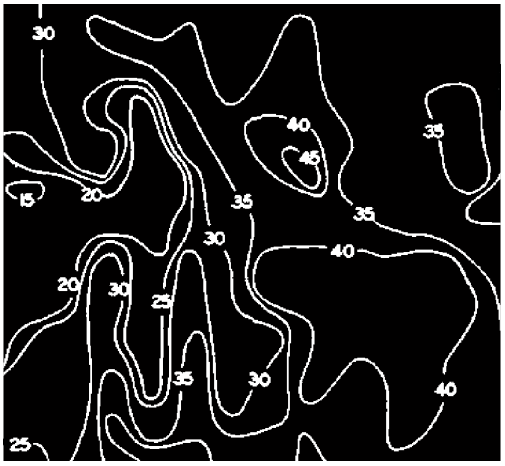
The algorithm is explained with the help of portion of image shown in red rectangle in Figure 3.11(a). The cropped image is shown in Figure 3.11(b).

3.2.1 Filter Text not Connected to Contours

Connected Component Analysis and Properties Extraction

Connected component analysis/labelling(CCA) is a basic and widely used technique in image processing and pattern recognition. Connected components labelling scans an image and groups its pixels into components based on pixel connectivity(4-connected or 8-connected), i.e. all pixels in a connected component share similar pixel intensity values and are connected with each other. Once all groups have been determined, each pixel is labelled with a gray level or a colour according to the component it was assigned to. Connected component labelling works by scanning an image, pixel-by-pixel (from top to bottom and left to right) in order to identify connected pixel regions, i.e. regions of adjacent

(a) Contour Layer from (JS07)



(b) Cropped Image

Figure 3.11: Sample Contour Image

pixels which share the same set of intensity values V, for a binary image V=1. (MS92)

When CCA is applied on the contour layer the components obtained can be classified into three categories:

1. Segments of Contour lines only
2. Altitude tags/Noise
3. Contour lines with the altitude tags attached

For example when CCA is applied on the portion of contour layer, 34 components are obtained which are shown in Figure 3.12(a). The three categories of components are shown in Figures 3.12(b), 3.12(c), 3.12(d) respectively.

In the first level, components are classified into altitude tags and contour lines using the region properties. Further classification between category 1 and category 3 i.e. contour lines with tags and lines without tags is explained in the next section.

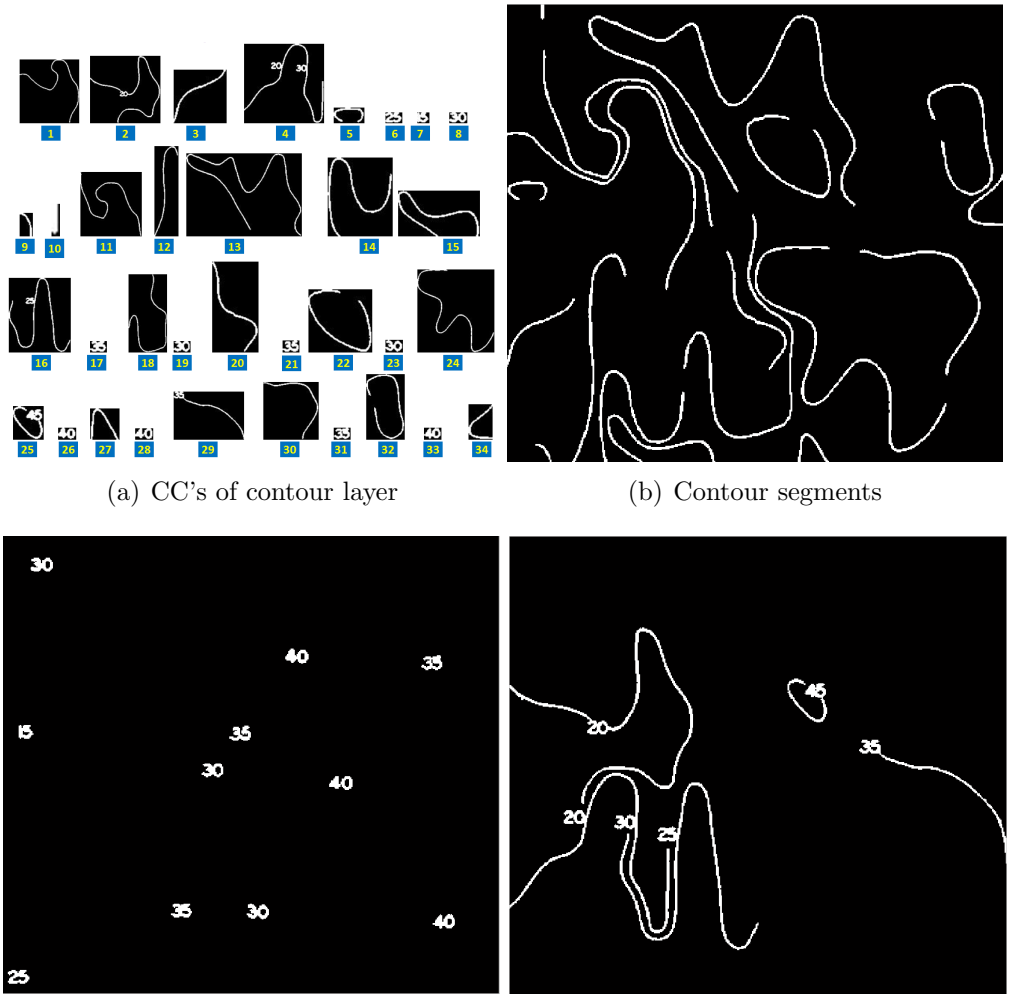
To filter the text not attached to the contours, i.e. components belonging to the second category, method proposed by Fletcher et al. (FK88) is employed. Each component is enclosed in a circumscribing rectangle (minimum bounding rectangle(MBR)) and the region properties of the component like MBR area, aspect ratio of MBR and the total object pixels are computed.

Table 3.5 lists the properties of each of the components shown in Figure 3.12(a).

Threshold can be arrived to filter the components belonging to text only by observing the histograms. From the data given in table 3.5, components 6, 7, 8, 17, 19, 21, 23, 26, 28, 31, 33 are classified using thresholds of 400, 2, 2 for area, aspect ratio and density respectively. Algorithm 3.5 explains the steps to filter text not connected to contours.

Figure 3.13(a) shows the filtered components, Figure 3.13(b) shows the image obtained after filtering text not connected to contours.

Once the text only components are filtered, altitude tags connected to contour lines have to be filtered, which is discussed in next section.



(a) CC's of contour layer

(b) Contour segments

(c) Altitude tags

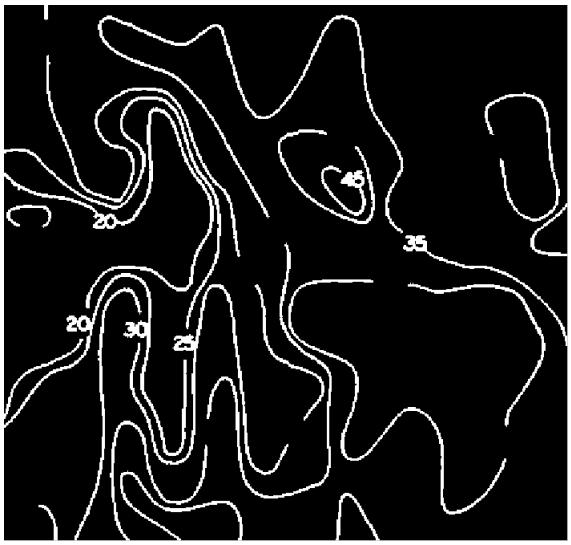
(d) Contours with attached text

Figure 3.12: Classification of Connected Components of Contour Layer

Table 3.5: Region Properties of Components

CC	MBR Area	Aspect	Density
1	33998	1.07	30.52
2	28380	1.04	20.11
3	4290	1.02	19.32
4	22500	1.00	15.77
5	703	1.95	4.29
6	260	1.54	1.51
7	182	1.08	1.43
8	273	1.62	1.55
9	464	1.81	5.21
10	42	4.67	1.05
11	26394	1.04	24.46
12	11385	3.76	18.45
13	45180	1.39	25.95
14	7840	1.23	13.18
15	5757	1.77	11.33
16	20410	1.21	15.27
17	260	1.54	1.53
18	21630	2.04	21.42
19	260	1.54	1.65
20	6384	1.96	17.88
21	280	1.43	1.68
22	6162	1.01	11.69
23	315	1.40	1.83
24	34547	1.08	30.25
25	1517	1.11	3.81
26	308	1.57	1.68
27	1404	1.08	6.47
28	286	1.69	1.71
29	12285	1.48	24.47
30	13334	1.04	23.98
31	280	1.43	1.68
32	6572	1.71	11.10
33	294	1.50	1.79
34	1290	1.43	6.94

(a) Components filtered



(b) Filtered Contour layer

Figure 3.13: Filtering Type I Text

Algorithm 3.5 FILTTEXTI(Img_c)

// Filter text not connected to contour lines //

Input:

Img_c:	Contour layer of topomap (Contour layer of L)

Output:

Img_{f1}:	Contour layer image after filtering Type 1 tags
$L_{t1},\ L_{t2}$:	Lists of components belonging to text and contours respectively.
$l_c,\ h_c,\ Area_c$:	Region properties

Var:

$MBR_c,\ MArea_c,$	Region properties
$Asp_c,\ Den_c$:	of component c
$T_{MArea},\ T_{Asp},\ T_{Den}$:	Thresholds on Area, Aspect ratio, Density
i:	Loop index

Method:

Find the connected components of contour layer, S_C denotes the set of components, $C_i, i = 1\ to\ n_c$
//Region properties computation of each component//
for $i \leftarrow 1\ to\ n_c$ **do**
  Find the Minimum Bounding Rectangle, MBR of C_i, assign to MBR_c
  Find the number of pixels of C_i, assign to $Area_c(i)$
  $l_c(i) \leftarrow length(MBR_c)$ //Computes length of MBR//
  $h_c(i) \leftarrow height(MBR_c)$ //Computes height of MBR//
  $MArea_c(i) \leftarrow l_c(i) \times h_c(i)$
  $Asp_c(i) \leftarrow max(l_c(i), h_c(i))/min(l_c(i), h_c(i))$
  $Den_c(i) \leftarrow MArea_c(i)/Area_c(i)$
end for
Find the thresholds on $MArea_c,\ Asp_c,\ Den_c$, assigned to $T_{MArea},\ T_{Asp},\ T_{Den}$ respectively
$Img_{f1} \leftarrow Img_c$ //Filter Type I(not connected) text//
for $i \leftarrow 1\ to\ n_c$ **do**
  if $(MArea_c(i) < T_{MArea})\ AND\ (Asp_c(i) < T_{Asp})\ AND\ (Den_c(i) < T_{Den})$ **then**
    Append i to L_{t1} //The components which belong to text are stored in list//
    Img_{f1} is updated by making area of image where component i is present equal to background
  else
    Append i to L_{t2} //Components which do not belong to text are stored in list//
  end if
end for
return $Img_{f1}, l_c,\ h_c,\ Area_c,\ L_{t1},\ L_{t2}$

end FILTTEXTI

Computational Complexity of FILTTEXTI*	
Step 1	$O(N^2)$
Steps 2-10	$n_c \times O(n)$ where n is number of pixels in each component
Step 11	If histogram computation, $O(N^2)$
Step 13-21	n_c
Algorithm Complexity: $O(N^2)$ where N is $Max(N_r, N_c)$	

*Steps with constant time complexity are not mentioned

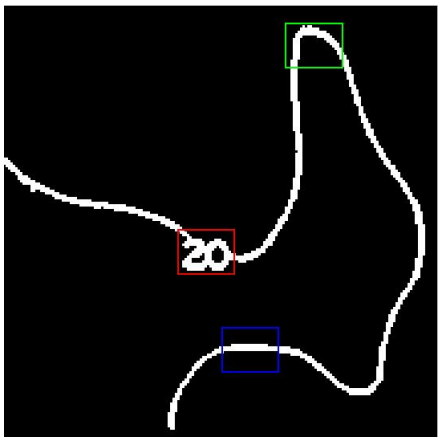
3.2.2 Filter Text Connected to Contours

The contour layer has altitude labels which are attached to it, as can be seen in Figure 3.13(b). As region properties fail to classify, a more local approach taking into account the spatial characteristics of the image is essential. Row and column projections of the binary regions have been used. Figure 3.14(a) shows one of the components of contour layer which has text attached. Different binary regions of the segment have been enclosed by three rectangles of same size, area enclosed in red is the text, area in green is the contour at a bend (two segments occur in the same window) and blue is a single contour segment. The row and column projections of the areas enclosed are shown in Figure 3.14(b) and 3.14(c). As it can be seen from the plots, the projections vary in three cases and hence can be used to differentiate text from contours in each component.

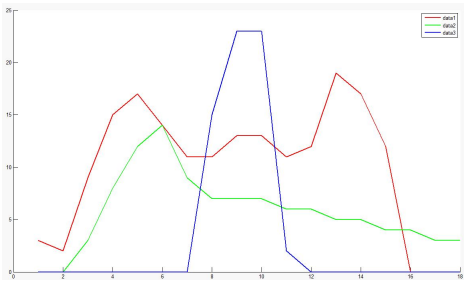
The area under the projection curve obtained from a window centred around an object pixel, is used to distinguish between contour and text pixels in a component. However two parameters are necessary to distinguish:

1. Size of window around the pixel where Projection has to be computed (w).
2. Threshold on the area under the projection curves i.e. total number of pixels in a window (T_{pc}).

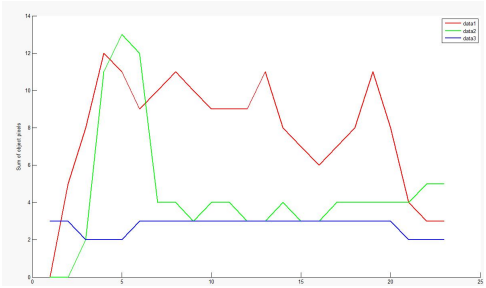
To arrive at both the above mentioned parameters, text(not connected to contours) components filtered from the image using Algorithm 3.5 are used. The length and height of the window are taken as the maximum of all the filtered MBRs dimensions (length, height) respectively. The threshold on the area is obtained from taking the average of total pixel counts of all the filtered components.



(a) Regions of Text and Contours



(b) Row Projections



(c) Column Projections

Figure 3.14: Row & Column Projections of Regions of Text and Contours

Algorithm 3.6 FILTTEXTII(Img_{f1}, l_c, h_c, $Area_c$, L_{t1}, L_{t2})

//Filter Text connected to contours //
Input:

Img_{f1}:	Contour layer after filtering Type I text
$l_c,\ h_c,\ Area_c$:	Region Properties of components
$L_{t1},\ L_{t2}$:	Lists of components belonging to text and contours respectively. All inputs are obtained from Algorithm 3.5)

Output:

Img_{f2}:	Contour Image after filtering altitude tags

Var:

$w_l,\ w_h$:	Window length and height
T_{pc}:	Threshold
n_{pc}:	Count

Method:

1: $w_l \leftarrow Max(l_c(L_{t1}))$ //Maximum length of all filtered text components//
2: $w_h \leftarrow Max(h_c(L_{t1}))$
3: $T_{pc} \leftarrow Average(Area_c(L_{t1}))$ //Average area of all filtered text components//

4: $Img_{f2} \leftarrow Img_{f1}$
//for each component belonging to contours//
5: **for** each component in L_{t2} **do**
6: The locations of all the object pixels in the component are stored.
7: **for** each object pixel p **do**
8: A neighbourhood of size $w_l \times w_h$, centred around p is considered.
9: Total count of the object pixels in the window, i.e. area under the row or column projection curve, is computed, assigned to n_{pc}
10: **if** $n_{pc} > T_{pc}$ **then**
11: p belongs to the text and Img_{f2} is updated by making pixel equal to background (zero in this case).
12: **end if**
13: **end for**
14: **end for**
15: **return** Img_{f2}
end FILTTEXT11

Computational Complexity of FILTTEXTII*	
Steps 1, 2, 3	Size of L_{t1}
Steps 5	Size of L_{t2}
Step 7	Total pixels of components in L_{t2} $: < N_r \times N_c$
Algorithm Complexity: $O(N^2)$	

*Steps with constant time complexity are not mentioned

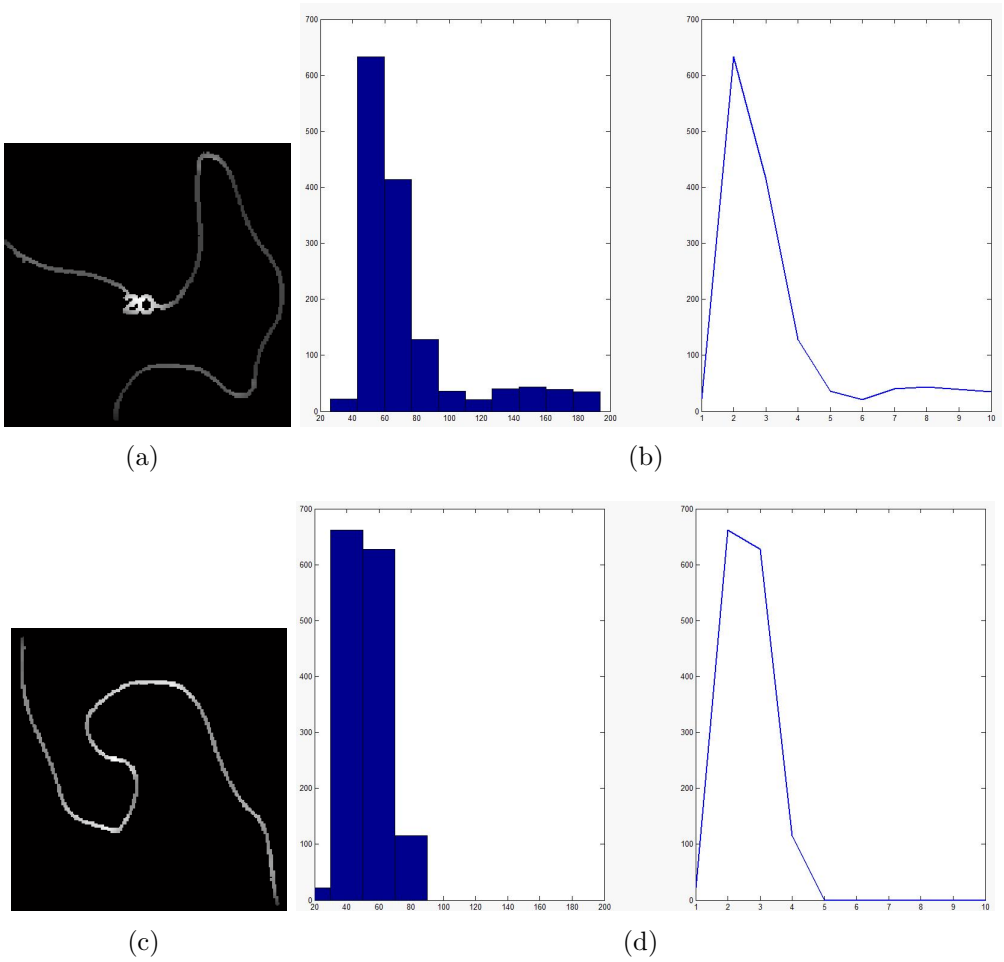


Figure 3.15: Projection Images of Text and Contours

Algorithm 3.6 explains the various steps involved in the filtering of contour layer from altitude tags attached to the contours.

The Figure 3.15 shows two images of contour lines with text and without text, where each pixel is replaced by the total count of pixels around a window, centred at that pixel. The figure also shows the histogram plot of object pixel intensities of the two images.

It can be seen in the Figure 3.15(a), that the pixels where text is present are brighter, and the histogram has values beyond a threshold, when compared to contour without text. The Figure 3.16 shows the pixels whose intensity value is greater than a threshold in red.

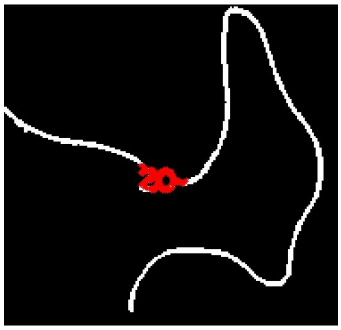


Figure 3.16: Pixels with Text (higher intensities)

Figure 3.17(a) shows the contour layer after employing filtering algorithms and 3.17(b) shows the altitude labels.

The algorithm works on the assumption that contour layer always contains tags which are not connected to any of the contour lines, and hence takes inputs from the Algorithm 3.5. However if the assumption fails for a contour layer, the algorithm needs inputs from the user.

3.3 Recognition of Altitude Tags

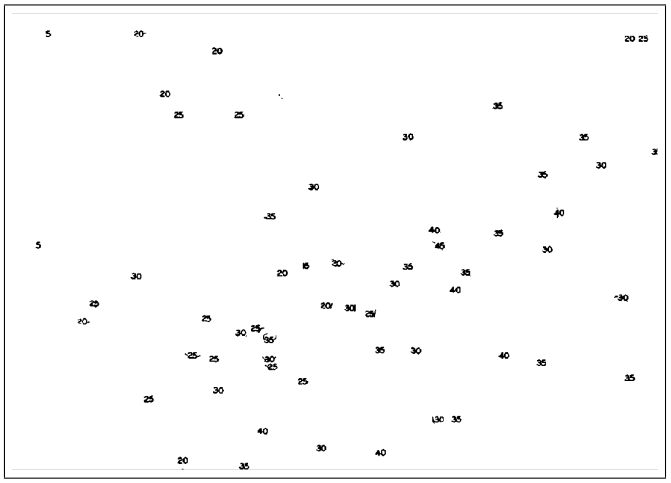
In the previous section, filtering of altitude tags of contour lines has been explained. These altitude tags are important as they convey the elevation information of the contour line. To obtain a three-dimensional surface from the contour layer, the altitude tag values have to be recognized and tagged to the contour lines. Altitude tag recognition is discussed in this section. Figure 3.18 shows the samples of altitude tags extracted from a topomap.

The following properties of tags in the topomap make altitude recognition a challenging problem:

1. Altitude tags do not belong to any standard font, the tags are not printed but hand written using markers.

2. Altitude tags are not aligned along any axis, but rotated along the contour line to which the value has to be tagged.

(a)



(b)

Figure 3.17: Altitude Tag Filtering

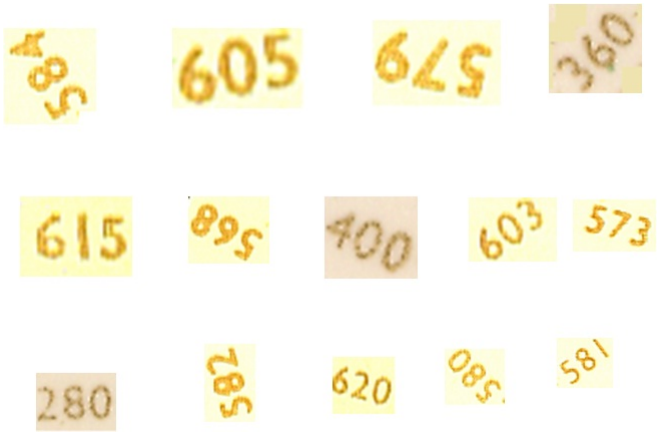


Figure 3.18: Sample Altitude Tags of Topomaps

3. Altitude tags are noisy. As text is extracted from a scanned colour topographic map, gaps are formed due to variations in colour introduced by the scanning process. Also when text attached to contour lines is filtered from the contour layer, some noise gets attached to the tag.

The altitude tag recognition is divided into three stages:

1. Stage I : Preprocessing
    - Separation of digits in the altitude tag
    - Size normalization
    - Rotation normalization
2. Stage II: Recognition of digits
3. Stage III: Altitude tag recognition
    - Recognition of the whole altitude tag
    - Validation

3.3.1 Preprocessing

1. Connected component analysis(CCA) is applied on each altitude tag to separate the digits. However this does not help when the digits in the tag

are connected as shown in the Figure 3.19(a). In the case of topographic maps, it can be assumed that the number of digits in the altitude tags across the map is same. Hence region properties of the tags can be used to segment the digits. When CCA does not yield the same number of components as the digits in an altitude tag, properties of each component like height, width are measured and and the component is split across the major dimension.

2. Morphological closing operation is performed to fill any holes in the digit as shown in Figure 3.19(b)

3. Each digit after segmenting, has to be rotated, so that all the digits are aligned along a common axis. Principal axis rotation has been used, i.e. the image is rotated along the first principal component axis, i.e along which the variance is maximum. The spatial co-ordinates of object pixels are found, principal components are computed, and then image is rotated at an angle of Cos^{-1} of first principal component. However the digits may be rotated by 180 deg, as shown in Figure 3.19(c), and 6 and 9 are incorrectly recognized.

4. The digit is normalized to a standard size. We have normalized the tags to a size of 28×28.

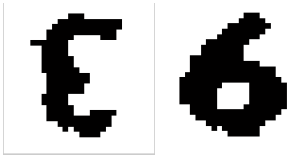
Figure 3.20 shows some digits obtained after the preprocessing on the altitude tags obtained from altitude tags shown in Figure 3.18.

3.3.2 Recognition of Digits

Handwritten digit recognition is an active topic in OCR applications and pattern classification/learning research. The literature on this topic alone is extremely huge, with a large variety of feature extraction and classification techniques being published every year. Features extracted range from geometric moments to contours and curvatures, while classification techniques range from template matching to neural networks (LNSF03). Databases like CENPARMI (Sue92), CEDAR (LS93), and MNIST (Y.L95) have been widely used in classifier design

(a) Connected digits

(b) Noise/Holes in digits



(c) Improper rotation

Figure 3.19: Problems with Extracting Digits from Altitude Tag

and evaluated in character recognition and classification/learning research. Every database is partitioned into a standard training data set and a test data set such that the results of different algorithms can be fairly compared. The MNIST (modified NIST) database was extracted from the NIST special databases SD3 and SD7, containing binary images of handwritten digits. SD3 and SD7 were released as the training and test data sets, respectively, for the First Census OCR Systems Conference. The database contains 60,000 handwritten digits in the training set and 10,000 handwritten digits in the test set. Some images are shown in Figure 3.21.

Different classifiers proved on this database by LeCun (YLYP98) had shown recognition rate from 88% till 99.3% (Table 3.6).

The convolutional neural network, LeNet - 5 created by LeCun et al. (YLYP98) has been used for the recognition of altitude tags. This class of networks have been successfully used in many practical applications, such as handwritten digits recognition, face detection, robot navigation and others. This network provides some degree of shift, scale and distortion invariance as their basic architectural

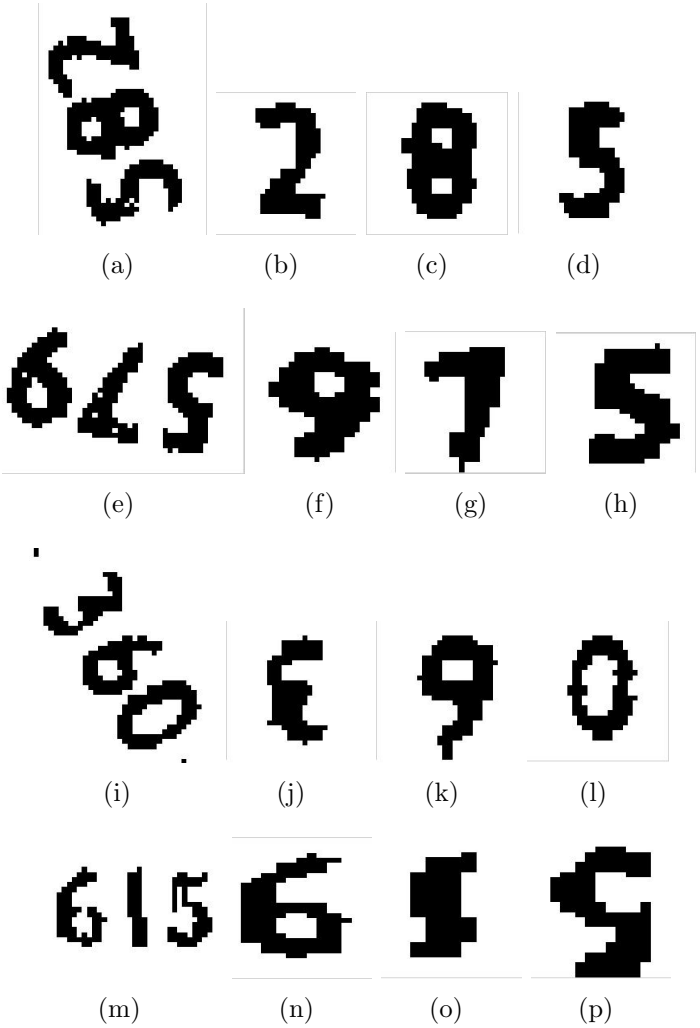


Figure 3.20: Sample Digits obtained after Pre Processing

Table 3.6: Recognition Rate of Different Classifiers

METHODS	ERROR RATE
Linear classifier	12.0
Linear classifier (nearest neighbor-NN)	8.4
Pairwise linear classifier	7.6
K-NN, Euclidean	5.0
2-layer NN, 300 hidden units (HU) (28x28-300-10)	4.7
2-layer NN, 1000 HU (28x28-1000-10)	4.5
2-layer NN, 1000 HU, [distortions] (28x28-1000-10)	3.8
2-layer NN, 300 HU, [distortions] (28x28-300-10)	3.6
1000 RBF (Radial Basis Function) + linear classifier	3.6
40 PCA (Principal Component Analysis) + quadratic classifier	3.3
3-layer NN, 300+100 HU (28x28-300-100-10)	3.05
3-layer NN, 500+150 HU (28x28-500-150-10)	2.95
3-layer NN, 300+100 HU, [distortions] (28x28-300-100-10)	2.5
3-layer NN, 500+150 HU, [distortions] (28x28-500-150-10)	2.45
K-NN Euclidean, deslant	2.4
LeNet-1 [16x16]	1.7
2-layer NN, 300 HU, [deslant] (20x20-300-10)	1.6
K-NN, Tangent Distance, [16x16]	1.1
SVM (Support Vector Machine) poly 4	1.1
LeNet-4	1.1
LeNet-4 / K-NN	1.1
LeNet-4 / Local	1.1
Reduced Set SVM poly 5	1.0
LeNet-5	**0.95**

Figure 3.21: Sample Images of MNIST Data

ideas include local receptive fields, shared weights (or weight replication), and spatial or temporal sub sampling.

Topology of LeNet - 5 (YLYP98)

The architecture of LeNet - 5, which is a typical convolutional neural network used for recognizing characters, is shown in Figure 3.22. LeNet-5 consists of eight layers. The input layer size is 32×32, even though the input images used are at most 20×20. so that the relevant features are then guaranteed to be contained in all feature maps and not get lost because they are near the boundary. The first convolutional layer has six feature maps, each of which has a resolution of 28×28, with a receptive field of 5×5. The second layer, or the first subsampling layer, contains six feature maps of size 14×14. The third layer is another convolutional layer and has 16 feature maps with size 10×10, with a receptive field of 5×5. The fourth layer contains 16 feature maps as well, each of which is of size 5×5. The fifth layer is a convolutional layer with 120 feature maps, again with a receptive field of 5×5. Then follows a layer with 84 neurons, which is fully interconnected with the previous layer. All neurons up to and including the sixth layer compute their input by calculating the weighted sum and feeding the result to the squashing function

$$f(u) = A\ tanh(Su)$$

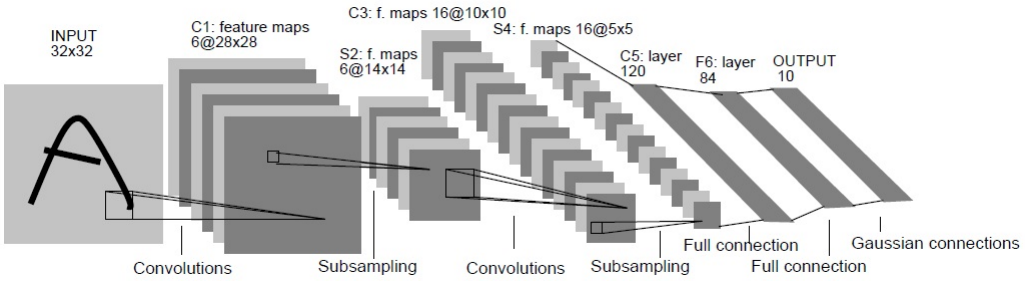
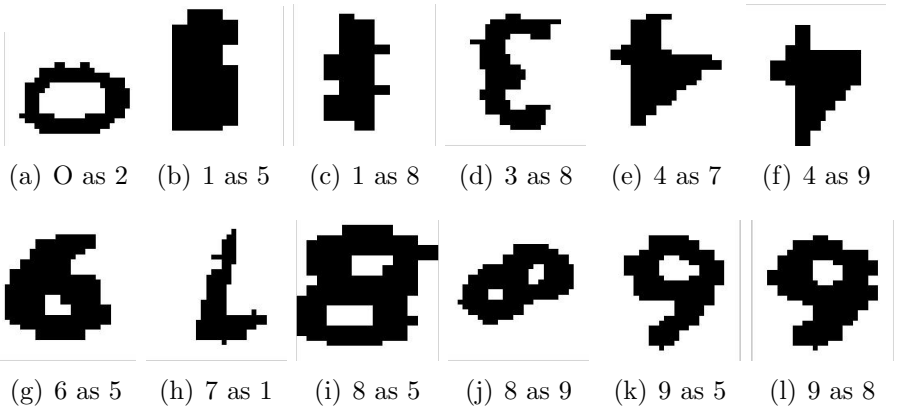


Figure 3.22: Architecture of LeNet - 5



(a) O as 2 (b) 1 as 5 (c) 1 as 8 (d) 3 as 8 (e) 4 as 7 (f) 4 as 9

(g) 6 as 5 (h) 7 as 1 (i) 8 as 5 (j) 8 as 9 (k) 9 as 5 (l) 9 as 8

Figure 3.23: Digits Misclassified

where A was chosen to be 1.7159, and S determines the slope of the function at the origin. Finally, the last layer contains ten radial basis function neurons.

3.3.3 Altitude Tag Recognition

The convolutional neural network is trained using MNIST database. The pre processed digits obtained from the previous step are normalized such that the mean is 0 and variance is 1. When tested on digits extracted from altitude tags of some topographic maps, shown in Figure 3.7, the accuracy is about 73%. Of the 106 digits, 29 have been misclassified, some of which are shown in the Figure 3.23. It can be noted that most of the errors in recognition are due to presence of noise or orientation of the digits. Each of the digits segmented from the set of altitude tags are recognized using LeNet - 5. The digits corresponding to a tag are grouped and the value of the whole tag is computed based on the place value

of the digit. However, the accuracy of recognition of digits is not 100%, and a post processing stage is essential to validate the recognition. The error rate of recognition can be reduced by adding knowledge to the system. The following rules of altitude tags are used in validating the recognized tags:

1. Altitude values are tagged to contour lines through out the topomap. Hence, same contour line has more than one altitude tag at different places. Altitude values tagged to the same contour line have the same value.

2. Altitude values change in fixed intervals (e.g. 5, 10 etc.) across the contour lines.

3. Altitude values of the intermediate contour lines which lie between two index contour lines lie in the range of the altitude values of the index lines.

The Validation stage has not been included in the thesis at this stage as it demands solutions to several issues regarding knowledge representation etc. However altitude value tagging to contour lines is addressed in the subsequent chapters.

3.4 Results

The Figure 3.24 shows the contour layers obtained from topomaps shown in Figure 3.7 extracted using the procedure explained. The first column is the extracted contour layer using clustering, second column is the filtered contour layer, and third column shows the filtered noise and text.

In the contour layer, shown in Figure 3.25, altitude values which belong to the first category (Section 3.2), are filtered using properties of minimum bounding rectangle. The other altitude tags are selected using projection images of connected components. Digits are extracted from the separated altitude tags and given as input to LeNet - 5 network. Of the total 112 digits segmented from the contour image shown in Figure 3.25, 61 have been correctly recognized. Of the 51 mis-classified digits, 19 errors are due to improper pre processing like segmentation of digits from the tag which leads to presence of noise or holes in the digit. All the fours(4) have not been recognized as they vary to a greater degree from

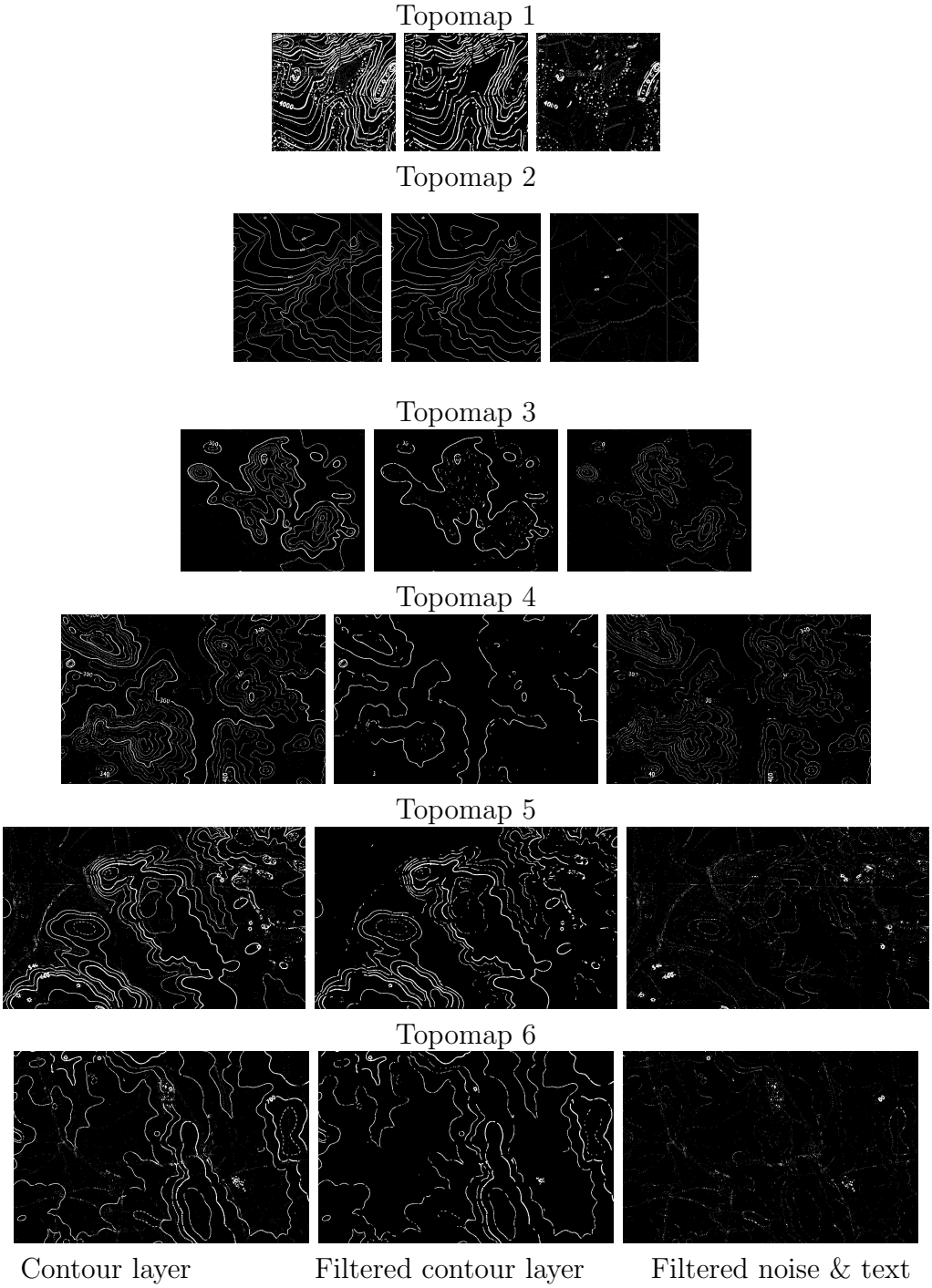


Figure 3.24: Filtering of Contour Layer

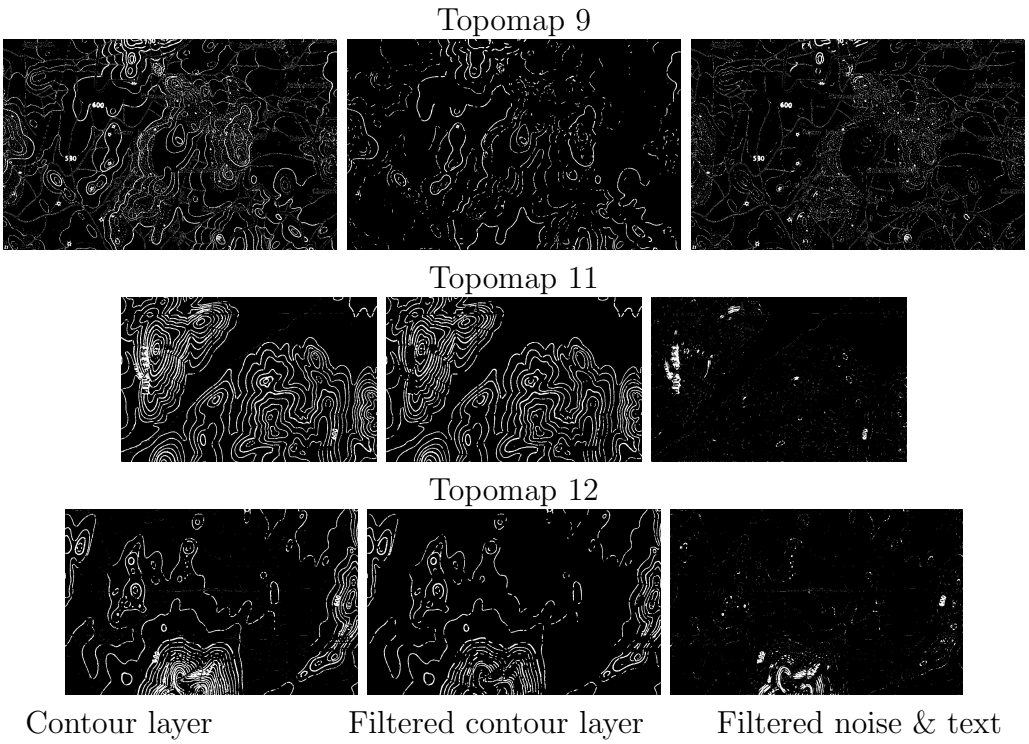
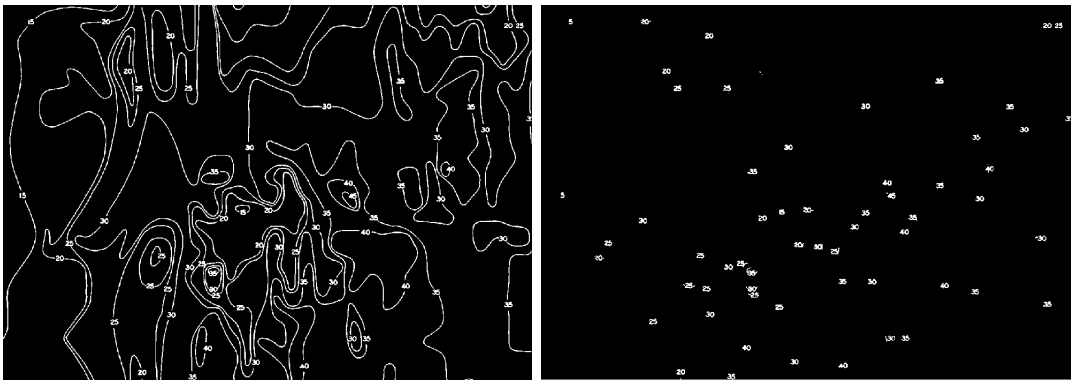


Figure 3.24: Filtering of Contour Layer



(a) Image from (JS07) (b) Filtered Altitude tags

Figure 3.25: Altitude Tag Filtering

Table 3.7: Altitude Digit Recognition

Digit	Total No	Errors due to Preprocessing	Errors in Recognition	Accuracy %
0	28	1	3	88
1	2	2	0	0
2	20	9	11	0
3	27	2	1	96
4	7	3	4	0
5	28	2	13	50

the standard training set of the network shown in Figure 3.21. Similarly 2's also have not been recognized at all. Hence from the remaining 93 digits, 32 digits have been wrongly recognized which amounts to an accuracy of about 66%. Table 3.7 shows the recognition of individual digits. As mentioned earlier this can be improved by adding knowledge about the system and by training the network using digits from the altitude tags of various topographic maps. For example, a contour line is tagged with a value of 30 at ten different places. However, out of 10 tags, two digits of a tag have been correctly recognized in 6 tags. The recognition errors in the remaining four tags can be corrected as we know that they belong to the same contour as the correctly recognized tag 30.

A colour topographic map is clustered into layers like contour, vegetation, water bodies etc. based on colour features. However the contour layer is subjected to further processing as the altitude tags which represent the elevation value of

the contour are also present along with the lines. An approach to filter altitude tags attached to the contours is discussed. It can be observed that the labels thus extracted can be further useful to tag the contour lines with the altitude values and have a 3D model. All the steps in the proposed approach are highly amenable to complex shapes of contour lines. The recognition of altitude tags using LeNet - 5 has been discussed. However recognition of altitude tags requires much more work as the reported techniques need to be further studied with larger training sets.

Chapter 4

Representation of Contour Layer

4.1 Introduction

When overlapping and intersecting layers (text, grid lines etc.) are removed, gaps are formed in the contour lines. Storing raw images of contour layer cannot be thus used for subsequent processing steps like vectorization, 3-D modelling. Since these curves are arbitrary in shape, they cannot be described by finite mathematical expressions. Hence an efficient, stable and robust representation scheme has to be formulated. Some properties of contour lines viz. two contour lines never intersect, contour line is always closed or ends at the boundary of the topomap can be utilized while arriving at a suitable representation. A good representation should have the following features:

1. It has to handle various complex operations like querying, change detection etc.

2. It should support visualization which involves various transformations like scaling, rotation. Hence it should support multiple resolutions and should be invariant to affine transformations.

3. Topomap images, owing to their large size, require a parallel computational architecture for processing. Therefore, a good representation of contour layer should be amenable to parallel implementation.

4. The representation has to be effective in addressing both region level and pixel level operations to be performed on the contour layer.

Survey on curve representation techniques shows that they can be classified into contour (boundary) based approaches and region based approaches. Some of these methods use a hierarchical representation to build approximations of curves at several levels-of-detail or resolutions.

In Polygonal approximation, a set of vertices are located on the given curve such that when they are joined by straight lines, a polygon that captures the shape of the curve is formed. This can be achieved by dominant point detection ((IJ84), (CR88), (BK92), (D.S93), (P.C97) etc.) or by a search process ((JE94), (P.Y99; P.Y03), (SY99), (M.S02) etc.). Some standard hierarchical representations of curves are: Arc tree (GW90), Strip tree (Bal81) etc. To achieve better performance than Arc tree, (SRS03) have introduced a new data structure called Equal error tree. Equal error tree outperforms Arc tree and compares well with Strip tree. (Gan06) has modified Arc tree to overcome the data compression related shortcoming. Equal error tree takes high complexity values, best case $O(n^2 logn)$ and worst case $O(n^3)$ which may limit its usefulness for large scale curves.

Common region based methods use moment descriptors to describe shapes. One such approach appears in (WY00) which uses Zernicke moments as a shape descriptor. Other region based methods include grid method, shape matrix, convex hull and median axis. (YMSA08) provided a new shape representation technique which utilizes the Hilbert space-filling curve. (AAC09) have expanded this method by providing a way to make the method rotation-invariant. Similar to the contour structural methods, region-based structural methods decompose the region into parts which are then used for shape representation and description. In the domain of hierarchical region representation, the Quad tree and Oct tree ((GK79); (I.G82); (Sam84)) have been proposed. Quadtree approach to region representation, termed a region quadtree, is based on the regular decomposition of the image into four equal-sized quadrants until each block is entirely contained in the region or entirely disjoint from it.

Contour shape descriptors are generally sensitive to noise and variations because they only use a small part of shape information, that is, contour information. Contour lines of topomaps are not clean, as they are extracted from a colour topomap, (challenges have been discussed in detail in (KZ03)), and hence methods like thinning or boundary following which aim at finding the boundary could result in loss of information. Hence region based approaches could be a better choice than boundary based methods, for contour lines. A Region-based structural method that decomposes each region into parts which are then used for shape representation and description has been employed. A hierarchical data structure is used for reasons explained earlier.

This chapter explains the representation of the primitive connected contour segment in such a way that it is easy to represent and reconstruct the contour line by combining the associated components. The chapter is organized as follows: In Section 4.2 we describe our proposed hierarchical contour layer representation. Section 4.3 explains how contour line is traced using the representation. Section 4.4 introduces the Relational Model, which is necessary to preserve the connectivity of represented contour lines. Section 4.5 discusses how the representation addresses some of the common queries on the contour layer. Section 4.6 illustrates the results of application of the algorithm on a contour layer extracted from topomap.

4.2 Tree Generation from Contour Line

Contour layer consists of contour lines, whose width depend on the scanning resolution of the topomap. For example, contour lines extracted from a topomap scanned at a resolution of 300 dpi, are 3-4 pixels wide. Each of these lines can be represented by different mathematical models like numerical interpolation, splines, Bezier curve etc. However as the shape of contour lines is complex, the curve is initially divided into smaller segments to minimize the error in approximating them. For this purpose, a region based convex hull approach has been adopted to segment the curve.

4.2.1 Contour Line Segmentation

Connected component analysis is applied on the binary contour layer obtained from Extraction stage of topomap processing (Ref: Img_{f2} from Algorithm 3.6). Each component of CCA gives one contour segment of the contour layer. The following procedure is applied to each contour segment/component of the contour layer, Img_l:

1. The contour segment is enclosed in a minimum bounding rectangle(MBR).
2. The ratio of number of pixels on a contour segment to the area of the enclosing Minimum Bounding Rectangle (MBR), called density ratio, is used as the measure of complexity of the shape of the contour segment.
3. A contour segment is split if the ratio is less than a specified threshold.
4. The direction of split is along the minimum dimension of MBR. In other words, the curve is split vertically if the length of MBR is more than or equal to the height and horizontally if the height is greater than length.
5. Steps 1 - 4 are repeated on each sub-segment of the contour, until the density ratio for all the sub-segments is more than the threshold value.
6. Steps 1 - 5 are repeated for each component of the contour layer.

The contour layer is hence divided into small sub-segments of contours.

4.2.2 Binary Tree Generation

A binary tree structure is used for hierarchical representation of the contour line. The tree is designed in such a way that it provides a hierarchical coarse to fine representation of the curve. The head node of the tree represents the whole contour line. The child nodes denote the sub-segments or parts of the contour represented by the parent node. Hence leaf nodes denote the final segments of the contour line. The steps for binary tree generation are as follows, keeping in mind three cases when a contour line is split.

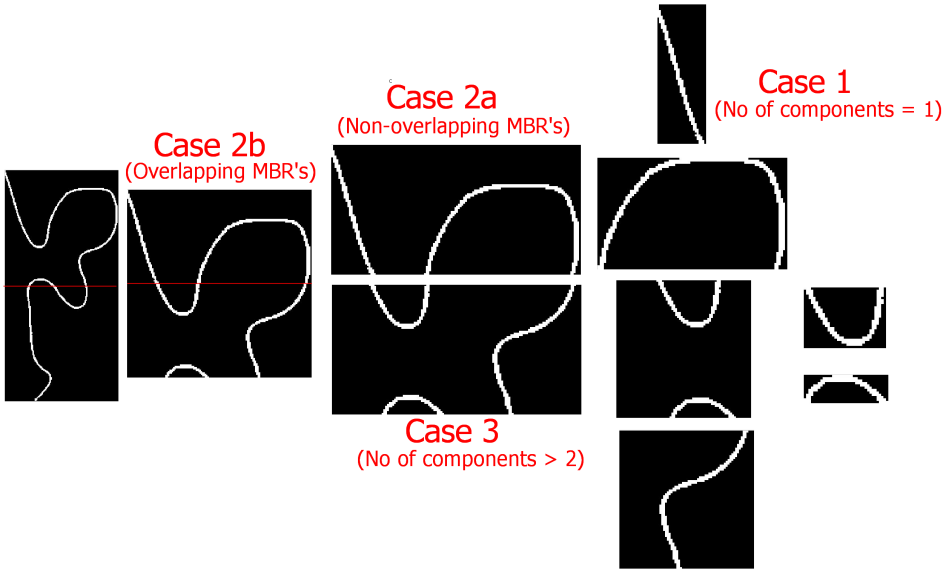


Figure 4.1: Cases which Occur when a Contour Line is Split

Case 1: Each split region has only one connected component (one contour segment) and thus only one MBR. This is the simplest case and requires computing the ratio of the curve to the area of MBR. If the ratio is less, splitting continues recursively or else the node data has to be updated to store the MBR of this partition.

Case 2: Each split region has two connected components and thus two MBRs. Two cases arise in such scenario

a) Non-overlapping MBRs: The node is split and the two MBRs are added as two children of that node and the procedure is repeated for the child nodes recursively.

b) Overlapping MBRs: A rectangle enclosing the two components is formed, and the process of splitting is continued.

Case 3: Each split region has more than two components. The node is split into two along the minimum dimension of parent MBR and the process of splitting the curve is continued.

The three cases are illustrated in the Figure 4.1.

Algorithm 4.1 SPLITNODE(Img_l, n_T)

//Recursive Tree Generation for a Contour Line //

Input:

Img_l:	Contour line (component of I_{f2} obtained from 3.6)
n_T:	Pointer to a structure denoting node of tree (Figure 4.3)

Var:

N_{cc}:	Number of connected components
I_S, $\{I_S^C, C = 1\ to\ N_{cc}\}$:	Set of connected components
M_S:	MBR data of segment
T:	Threshold on density ratio

Method:

1: **if** $n_T.MBR$ is null **then**
2: $M_S \leftarrow$ MBR of Img_l //n_T is the head node//
3: **else**
4: $M_S \leftarrow n_T.MBR$
5: **end if**
6: $I_S \leftarrow$ portion of contour enclosed by M_S
7: Find the connected components of image, I_S, $\{I_S^C, C = 1\ to\ N_{cc}\}$
8: **if** $N_{cc} = 1$ **then**
9: $M_{c1} \leftarrow$ MBR of I_S
10: $dens\_img \leftarrow$ density ratio of I_s
11: **if** $dens\_img < T$ **then**
12: Split M_{c1} along the minimum dimension to obtain 2 MBR's, $M_S^1,\ M_S^2$
13: **else**
14: $M_S^1 \leftarrow M_{c1}$
15: $M_S^2 \leftarrow null$
16: **end if**
17: **else if** $N_{cc} = 2$ **then**
18: $M_S^1 \leftarrow MBR\ of\ I_S^1$
19: $M_S^2 \leftarrow MBR\ of\ I_S^2$
20: **if** $M_S^1 \cap M_S^2 \neq null$ **then**
21: $M_U \leftarrow M_S^1 \cup M_S^2$
22: Split M_U along the minimum dimension to obtain new, $M_S^1,\ M_S^2$
23: **end if**
24: **else**
25: Split M_S along the minimum dimension to obtain new $M_S^1,\ M_S^2$
26: **end if**
27: **if** $M_S^2 \neq null$ **then**
28: Update the tree by adding a left child to the node n_T
29: Update the tree by adding a right child to the node n_T
30: SPLITNODE(I_{inp},$n_T.left$) //Repeat with image in left child of node//
31: SPLITNODE(I_{inp},$n_T.right$) //Repeat with image in right child of node//
32: **else**
33: $n_T.MBR \leftarrow M_S^1$ //Update the node//
34: **end if**

end SPLITNODE

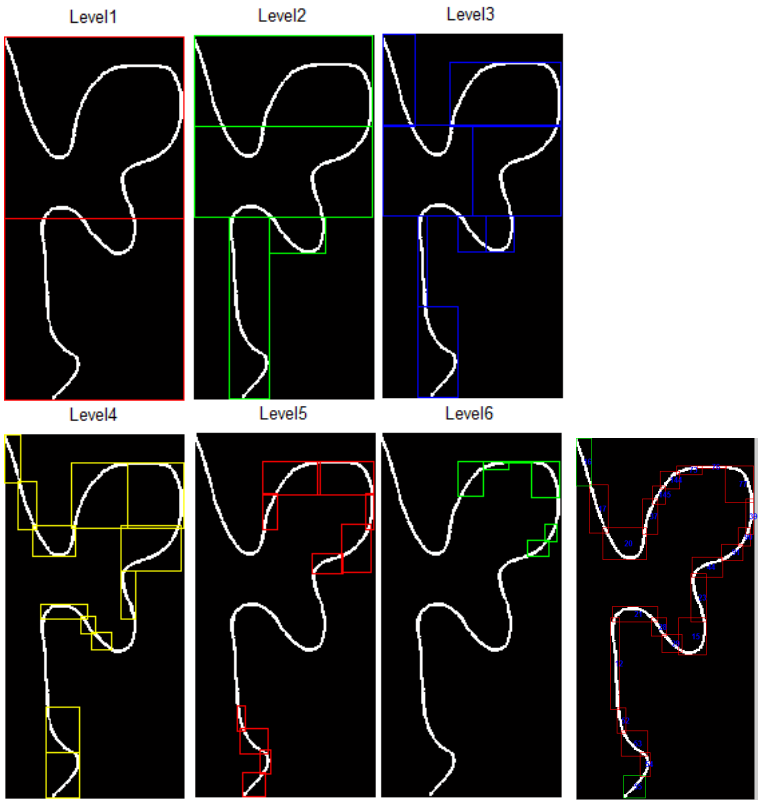
Computational Complexity of SPLITNODE
Recurrence relation :
$T(n) = \begin{cases} a & n = 1 \ AND \ dens\_img > T \ , \quad \text{a: constant} \\ 2T(n/2) + cn & \text{otherwise} \quad \text{c: constant} \end{cases}$
where n = number of segments of contour
When n is power of 2, solving by successive substitutions (HS98),
$T(n) = O(nlogn)$

The root of the tree stores the MBR coordinates. Two nodes are added to it when a curve is split, each having the coordinates of the bounding rectangle of respective curves. The procedure is repeated recursively for all the nodes. Algorithm 4.1 shows the pseudo code of the recursive tree generation algorithm. Figure 4.2(a) shows segments of contour line at each level in the tree, shown in Figure 4.2(c). Figure 4.2(b) shows the segments of curve obtained from the leaf nodes of the tree, which is shown in Figure 4.2(c). From the Figure 4.2(c), node 1 represents the whole curve, nodes 2 and 3 which are child nodes of 1 denote two segments shown in 'Level 1' of the curve. Each of these nodes are again split into two nodes and hence level 2 has four nodes i.e four segments as shown in Level 2 of Figure 4.2(a). The process is continued, till the contour line has been split into 23 segments. The tree has six levels, in which nodes at deeper levels represent smaller segments which are parts of segments belonging to nodes at higher level.

The contour line is finally represented as a binary tree in which all the leaf nodes store the MBR coordinates of the smaller segments in the curve. The entire contour line can be approximated by traversing only the leaf nodes which point to MBRs of individual segments. Hence each contour line of the contour layer is represented as a binary tree. However it is to be noted that leaf node traversal of the tree does not necessarily generate contiguous segments of curve, and therefore the curve cannot be traced. To overcome this problem a pointer is stored in the leaf nodes pointing to the next leaf node to be traversed. This has been explained in detail in the next section.

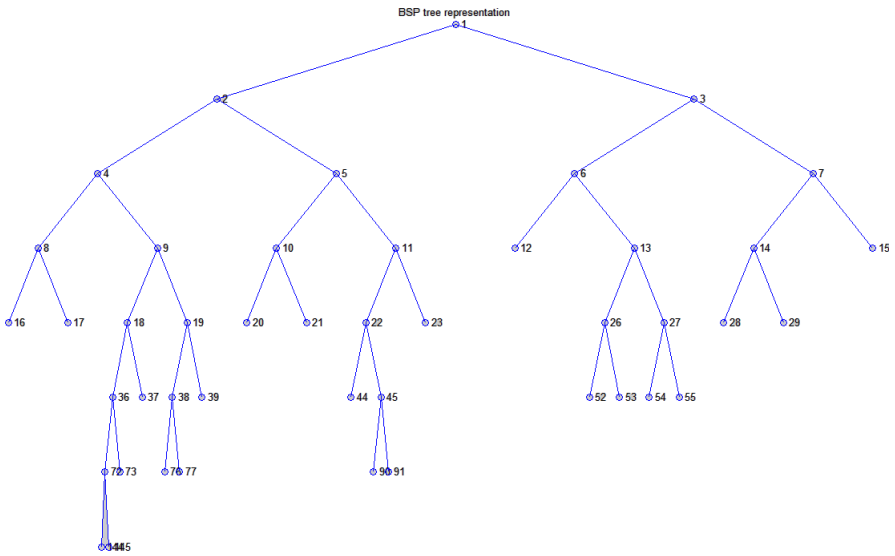
4.3 Modifying Tree using Threads

As a consequence of the previous step a tree is formed for each contour line in the contour layer. The depth first traversal of tree shows the contour segmen-



(a) Segments of Contour Line at each Level in Tree

(b) Segments corresponding to Leaf Nodes



(c) Tree generated from given Contour

Figure 4.2: Contour Line Segmentation

tation, starting from the contour line to the final segments. However, the tree does not preserve the order in which segments have to be traversed to trace the curve starting from one of the end points. Automatic tracing of a thick curve, is an involved and complex problem. Our algorithm addresses the problem by employing the tree based representation of a contour line.

4.3.1 Tracing Contour Line using Tree

The tracing algorithm works on the fact that the MBRs of two consecutive segments of the curve will overlap. Therefore, from segmentation of curve it is apparent that the end-segments of the curve have a single intersection with only one segment unlike all other segments which have multiple intersections. Let L denote the set $\{L_i, i = 1\ to\ N_l\}$ of leaf nodes.

The Algorithm 4.2 returns the list of leaf nodes, to be traversed to trace the contour from one end point to the other. In the case of an open contour, the list returned by the algorithm begins and ends with nodes representing end segments. In the case of closed contour, the list begins and ends with the same leaf node.

4.3.2 Updating of Tree: Adding Threads to Leaf Nodes

The information on the order of leaf nodes to be traversed to preserve the connectivity of segments is incorporated in the binary tree with the help of a pointer added to the leaf node, which is termed as thread. Hence the tree is modified by adding threads to the leaf nodes which point to the next and the previous leaf nodes to be traversed so that the curve can be traced by starting from any one of the end segments.

The fields in each node of the tree are shown in Figure 4.3. The head node of the tree, as shown in Figure 4.3(a), stores the absolute position of MBR enclosing the contour in the topomap, pointers to the leaf nodes which store the end segments of the contour, and attributes of the contour like altitude value etc. apart from the pointers which point to the child nodes. The position of top left corner, length and height of MBR are stored in the MBR field of the node. The Intermediate nodes and the Leaf nodes, shown in Figure 4.3(b), 4.3(c) respectively, store only relative position of the MBR with respect to the head node,

Algorithm 4.2 TRACECONT(L)

//Obtain order of leaf Nodes to trace the contour//
Input:
L: Set $\{L_i, i = 1\ to\ N_l\}$ of leaf nodes.
Output:
S: List of nodes required to be traversed in order to trace the curve
Var:
L_i^I: Set of leaf nodes intersecting with node L_i
N^I: Set of leaf nodes intersecting with node N

1: Start randomly with any leaf node L_i of the tree. Let S, which denotes the list of nodes required to be traversed in order to trace the curve, be initialized to L_i.
2: Search for the set L_i^I of leaf nodes $L_j \in F, \forall j \neq i$ such that MBR stored in node (L_j) intersect with MBR stored in node (L_i).
3: If L_i represents an end segment, $MBR(L_i)$ will be intersected by the MBR of only one other node, N and size of the set L_i^I is one. //If Algorithm 4.1 is followed for tree generation then in the above algorithm, the cardinality of L_i^I is at most 2. If cardinality of L_i^I is one, ith component represents an end segment.//
4: If L_i does not represent an end segment, size of the set $L_i^I > 1$ and any one of nodes in the set L_i^I can be chosen to be the next node, N in the list, S. N is removed from F
5: Steps 2 and 3 are repeated for N. If N does not represent an end segment, size of N^I is ≥ 1; otherwise set N^I is an empty set.
6: If the starting node L_i represents an end segment all the nodes will be added to the list S.
7: However if the starting node does not represent an end segment, the algorithm will terminate at the node representing an end segment, as for such a node there will be no other node with an intersecting MBR.
8: In such a case, the order of the nodes in the list is reversed. Thus the node representing the end segment will become the first entry in the list and the first entry(L_i) will be the last entry. Steps 2 - 4 are repeated with the reordered list
9: **return** S

end TRACECONT

Computational Complexity of TRACECONT
In the worst case, first node has to search $N_l - 1$ nodes, second node $N_l - 2$ leaf nodes etc. hence the complexity is of order $O(N^2)$ where N is number of leaf nodes/segments

Last LN	Left child	MBR (Absolute)	Attributes Altitude etc	Right child	First LN

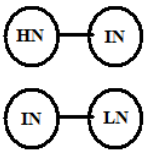
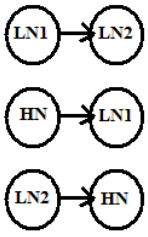
(a) Head Node Structure (HN)

Left child	MBR (Relative)	Data (Null)	Right child

(b) Non-Terminal / Intermediate Node Structure (IN)

Left child	MBR (Relative)	Data	Right child

(c) Terminal / Leaf Node Structure (LN)

	Denotes level of abstraction. IN is part of HN. LN is part of IN.
	Denotes order of connectivity. LN1.end = LN2.start. Curve.start = LN1.start Curve.end = LN2.end

(d) Edges in the Tree

Figure 4.3: Node Structure of the Binary Threaded Tree

enclosing that portion of the segment. These nodes also store a pointer to data, which store the data of contour segment. Nature of data depends on the method used to approximate the segment like chain codes, Bezier interpolation etc. The Intermediate node has null pointer in the data field whereas leaf nodes point to data.

The tree has two kinds of edges as shown in Figure 4.3(d). One edge denotes level of abstraction and the other is a thread which denotes order of connectivity. Hence traversing leaf nodes of the binary threaded tree gives the order of leaf nodes to be traversed to trace the contour.

An example of such tree generated for the curve shown in Figure 4.2(b) is shown in Figure 4.4. The starting leaf node is pointed by the head node and the final leaf node points back to head node. Leaf Nodes are linked, shown by blue arrows in the figure.

The explanation of the proposed tree representation covers only MBR's enclosing a segment of the contour and does not handle any information about the contour. This is further discussed in contour approximation in Chapter 6. This representation is useful in addressing certain region level processing requirements as will be discussed later.

4.4 Relational Model

4.4.1 Need for Model

Each connected component in a contour layer is represented as a binary threaded tree. From each tree, a list of leaf nodes which can trace the contour is computed. If the topological characteristics of contour lines (Section 1.2) are preserved , the entire contour layer is represented by binary threaded trees of each contour line. However due to improper segmentation and removal of overlapping/intersecting features contour lines are broken. Reconstruction of contour layer, discussed in Chapters 5, 6, handles operations required for closing broken contour lines (gaps). Such information about which contour lines are joined to form a closed contour should therefore be stored.

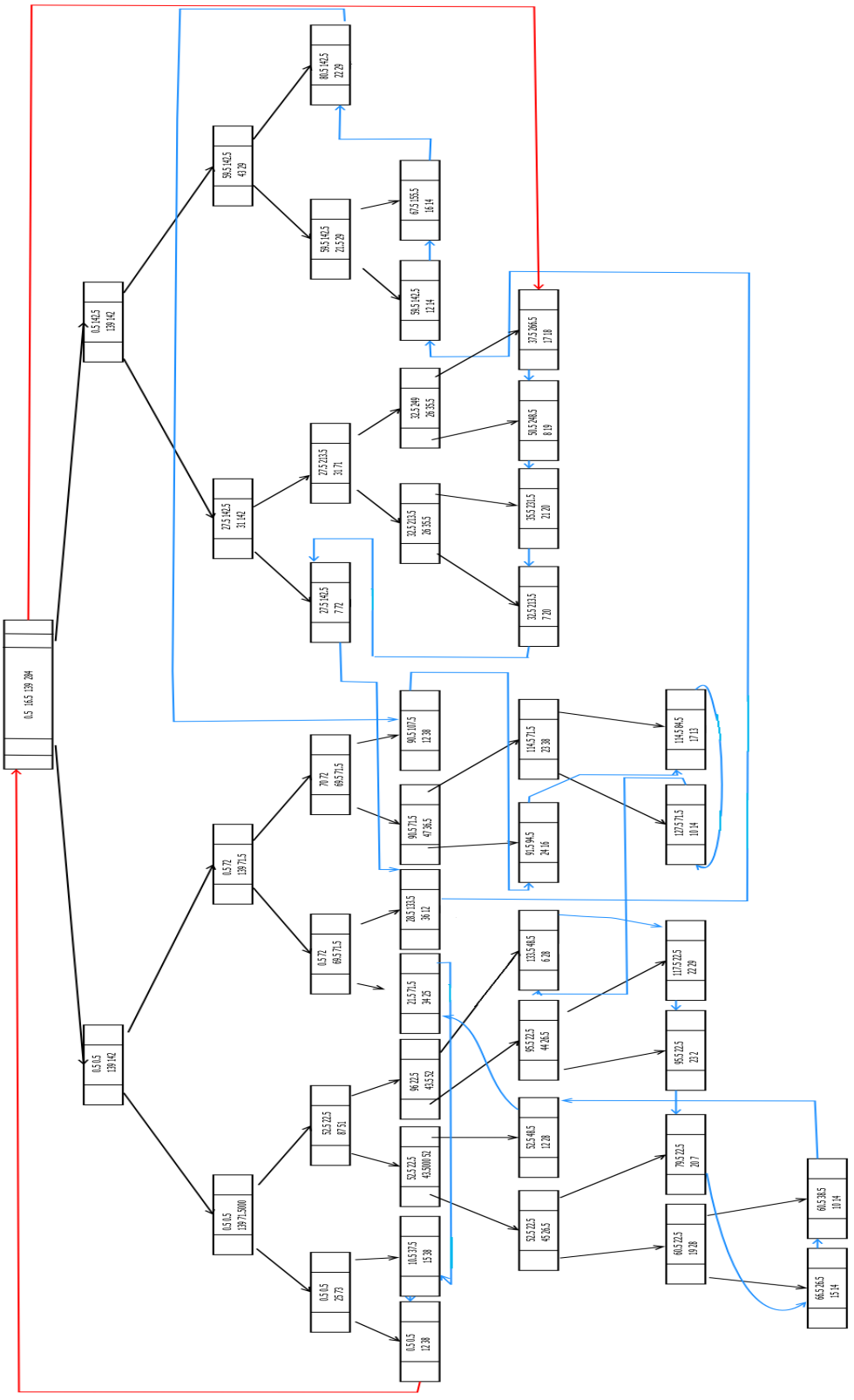


Figure 4.4: Threaded Binary Tree

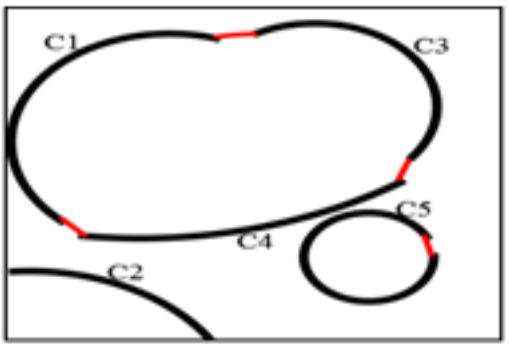
Another scenario where a relation between contour lines has to be maintained is when parallel architecture is used and parts of topomap are distributed across various nodes. The contour lines in each of these parts are represented using trees and the connectivity of different segments spread across different nodes has to be stored. Hence a relational model, which stores the connectivity between various contour lines (which have been represented using binary trees) is essential to represent the entire contour layer.

4.4.2 Generation of Relational Matrix

To indicate which two trees or equivalently the associated contours are connected, a relational model is defined between them. Such a model can be stored using various data structures like graphs, B-trees etc. An adjacency matrix has been used to represent such graphs. It is to be noted that a gap between two contours can be filled by four possible ways, considering the fact that the start and end points of two segments can be joined in four different ways i.e from start to start, start to end, end to start and end to end. A weighted directed graph is used for the representation. Weights 1, 2, 3 and 4 are added to the edges, each weight representing one of the four possible ways of joining the contours.

Let the number of connected components (CC) in a contour layer be N_{cc}. Each of these contours has been represented using a binary threaded tree. The associated adjacency matrix is of size $N_{cc} \times N_{cc}$. Let C_x and C_y be the two contours that are joined. The $(x, y)th$ element of the adjacency matrix has a non zero value (1 - 4). Note that the value of $(y, x)th$ element is non zero only if C_y is joined with C_x, if both $(x, y)th$ and $(y, x)th$ elements are non zero, C_x and C_y form closed contour. If a contour C_x has a discontinuity and is filled, the value of $(x, x)th$ element is non zero which indicates that it is a closed contour. Subsequently, employing back tracking on the matrix, a list of contour segments to be traversed to trace the entire contour line across the topomap is obtained. When all the components of matrix are listed, each segment appears only once i.e. there are a maximum of N_{cc} entries in the matrix.

An example of such matrix for the contour layer shown in Figure 4.5(a) is constructed. $C1$ to $C5$ are the connected components in the contour layer. Each



(a) Sample Contour Layer

	C1	C2	C3	C4	C5
C1	0	0	3	0	0
C2	0	0	0	0	0
C3	0	0	0	4	0
C4	1	0	0	0	0
C5	0	0	0	0	2

(b) Adjacency Matrix

Figure 4.5: Relational Model

contour line is represented as a binary threaded tree. Gaps are filled between segments $C1$, $C3$ and $C4$ (shown in red). $C5$ is closed when the gap is filled. The adjacency matrix is formed by placing weight 1 at position (1,4) as end of $C4$ is joined with start of $C1$, 2 at (5,5) as start of segment $C5$ is connected to end of segment $C5$, 3 at (1,3) as end of $C1$ is joined to end of $C3$, and 4 as start of $C3$ is joined to start of $C4$. The matrix is shown in the Figure 4.5(b). By using backtracking algorithm on the matrix, three contours are obtained which are:

- $C1(start->end), gapfill(C1.end, C3.end), C3(end->start),$ $gapfill(C3.start, C4.start), C4(start->end), gapfill(C4.end, C1.start)$
- $C5(start->end), gapfill(C5.start, C5.end)$
- $C2(start->end)$

It can be noted that this representation can be extended to the entire contour layer of the topographic map. Even when the map is divided among the various nodes of the parallel architecture, the same concept of Relational matrix is used to combine the results of the various nodes. Hence the entire contour layer is denoted as a forest, with each contour line being represented as a tree. It is interesting to extend the same concepts to other layers of topomap.

4.5 Query Processing

The contour lines extracted from a topomap have been represented using a threaded binary tree. In this section the scheme of handling various operations on the contour layer by the binary threaded tree is discussed. One of the operations on the contour layer is identification and filling of gaps introduced during extraction of the contour layer from a topographic map. Other operations are Vectorization, Attitude value tagging and 3D model generation. However these operations require several computations to be performed on the contour layer. In addition, from a users perspective, contour layer is useful in answering queries regarding the altitude values of point or region. Hence the representation of contour layer should be able to address a range of queries which are described below. Some of these queries are demonstrated using the image in Figure 4.6(a). Each contour is represented using a binary tree as shown in Figure 4.6(b),4.6(c),4.6(d) etc for C1, C2, C3 etc. respectively. Given below are some of the operations on contour layer:

1. Topological relationships between contours : This pertains to getting information on how two contours are related i.e. if a contour is within a contour or parallel to other etc. If two lines are not containing each other, the possibility of them being parallel can be considered by comparing the distance between them. A raster contour image has to be considerably processed to find out if a contour line exists inside another contour, or if two contours are neighbours. However with the tree representation, it is possible to get the MBR coordinates of the contours from the various head nodes. From the MBR coordinates, it is possible to find out if a contour contains another contour line. It can be seen from the Figure 4.6(a) that three contours C2, C3 and C4 are closed, C2 containing C3 and C4 whereas C1 and C5 are open contours. For example T2 head node stores [70 69 405 541] and T3 stores [190 145 200 281]. Clearly the MBR of T3 exists inside the MBR of T2. Therefore it can be concluded that C2 contains C3 because of the property that no two contour lines intersect. Similarly, comparing T1 and T2, the MBRs intersect and hence the contours are not concentric. Hence

the trees are traversed and the MBRs can be compared from the leaf nodes to check if they are parallel throughout their length.

2. Spatial query analysis : This is related to answering queries based on the spatial properties. For instance, to find out the approximate altitude range of a point search can be initiated from the head nodes. In the set containing contours, search can be made till the leaf nodes. This type of queries are explained in detail later.

3. Database mining: Apart from the queries mentioned above, when a database of contour layers of various topomaps is prepared and stored, some of the queries based on mining can be addressed. Given a portion of contour layer, a relevant issue that needs to be addressed is: " Can the database be searched and the layer containing the queried region extracted?". As the contours have been represented using trees and graphs, can this be addressed using Graph mining techniques?

4. Change Detection: This is concerned with comparison of two different contour lines to identify any similarity between them.

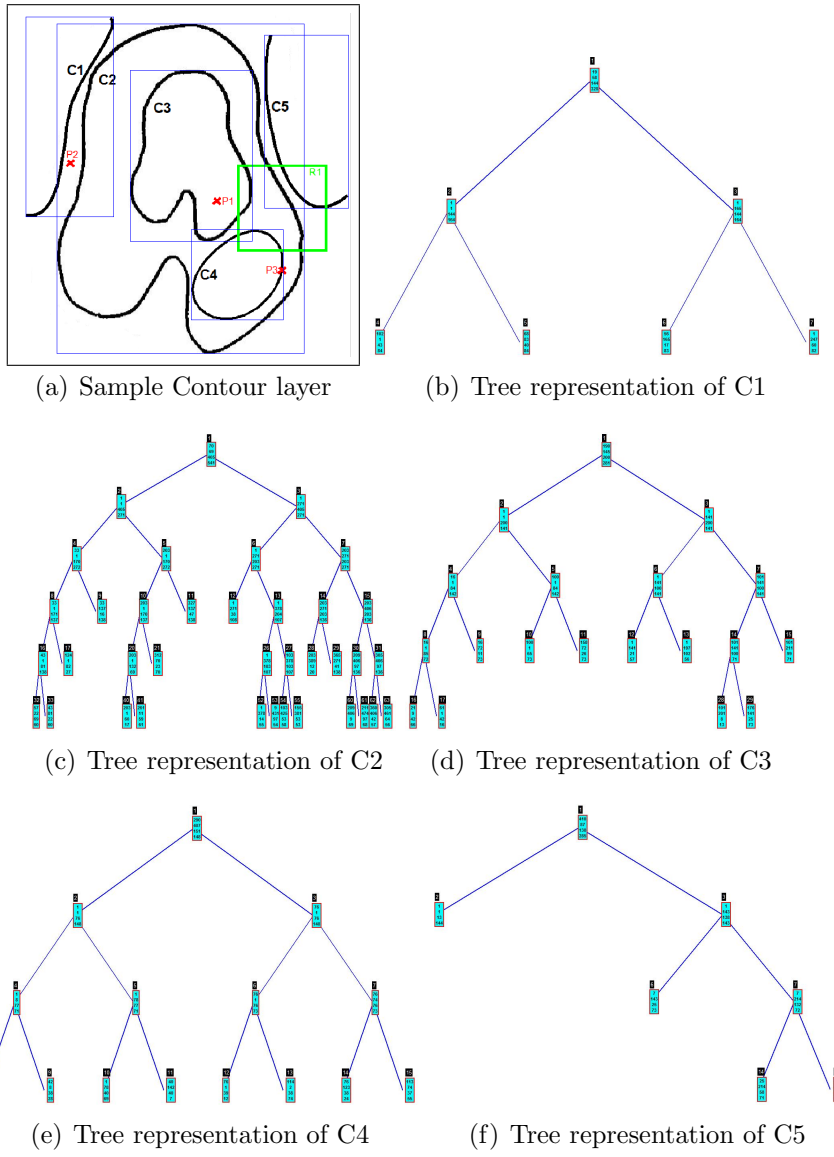
Spatial Query Analysis

Some of the spatial queries on contour layer are described below. These are broadly classified using the spatial query classification given by (BHKS93).

- Modifications
  This includes operations for insertion, deletion and update of information. These operations can be performed on the contour layer using the Relational model.

- Selections

  (A) Point Query: Given a query point P and a set of objects M, the point query yields all the objects of M geometrically containing P.
  Example Q1 : The nearest contour line to a point.



(a) Sample Contour layer (b) Tree representation of C1

(c) Tree representation of C2 (d) Tree representation of C3

(e) Tree representation of C4 (f) Tree representation of C5

Figure 4.6: Query Processing

This is useful in obtaining the altitude value at that point. Any contour can be related to the point in three ways. The point has to be inside the contour, outside the contour or on the contour. As the tree in its original form stores only MBRs of contours the query is addressed with respect to the containing MBR of the point. Given spatial coordinates of P, it is required to find all the contours, for which the MBR's represented in the respective head nodes contain point P. This can return more than one MBR and two cases arise in such a situation: The MBR's are concentric i.e. one contains another or they are intersecting.

Case 1: In the first case, the smallest MBR is chosen as the result and hence the point is in that region. In the Figure 4.6, point P1 is an example of this case. Point P1 is contained in C2 and C3. However as C2 contains C3, C3 is the closest contour to point P1.

Case 2: The second case requires more processing and hence the trees have to be traversed/searched to conclude as to which contour the point is nearest. For example consider point P2. Contours C1 and C2 are selected as their MBR's contain the point. Upon traversing the trees it can be concluded that the P2 is closer to node 9 of T2 and node 4 of T1 and does not actually exist on any of the contours.

(B) Region Query: Given a polygonal query region R and a set of objects M, the region query yields all the objects of M sharing points with R. A special case of the region query is the window query. The query region of a window query is given by a rectangle. Both, the window query and the region query are often called range queries.

Example Q2: What is the range of altitude values of selected region or the range of altitudes through which a river flows. To answer this query it is necessary to know all the contours which are closest or pass through the region. For example given region R1 in Figure 4.6(a) contours C2, C3, C4, C5 can be obtained as results as their MBR's intersect with that of R1.

- Combinations
  - Spatial Join: For two given object sets A and B the spatial join operation yields all pairs of objects whose spatial components intersect. More precisely, for each object we have to look for all objects in B intersecting with A. Note, that for efficient processing of the spatial join a selective spatial access to the objects is necessary.
    Example Q3: Given a contour, find the various regions through which this passes. For example, the map is divided into 8 regions (A1 to A8). To find through which regions C2 passes, it is important to trace the contour. The leaf nodes are traversed and the MBRs of regions are compared to MBRs of leaf nodes to obtain the intersecting regions.

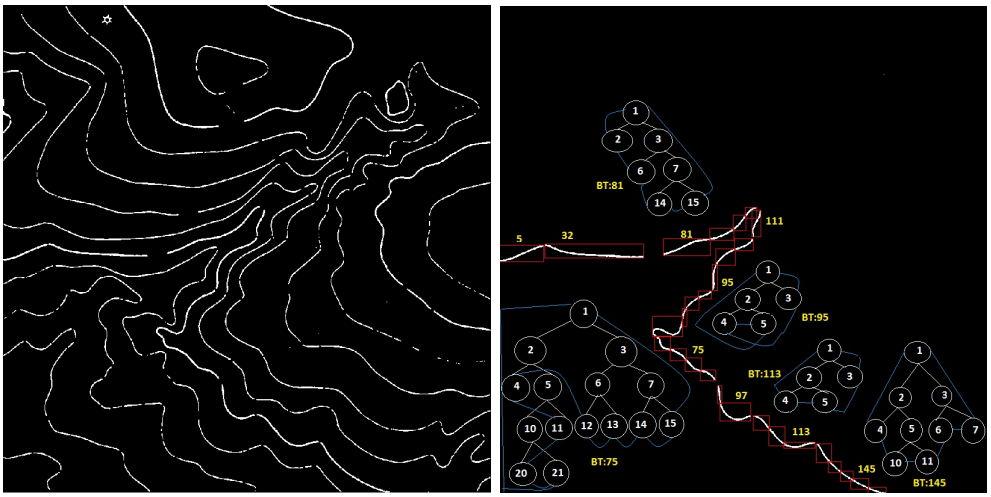
The utility of representation in addressing certain queries has been discussed. Our research not being specifically focused on spatial databases and query processing, but illustrates efficacy of our representation.

4.6 Results

The contour layers obtained from the 12 topomaps shown in chapter 3 (Figure 3.22) have been represented using binary trees. Figure 4.7(a) shows the contour layer of topomap 2, and Figure 4.7(b) shows few components of a contour line, the segments into which it has been divided, and the binary threaded tree of the components. Table 4.1 lists the number of connected components in each contour layer, number of segments into which these components have been divided and maximum depth of all the trees used to represent contour lines in each of the contour layer.

The proposed method has been applied on the contour layer, shown in Figure 4.8.

The image has 30 contour segments, therefore 30 pointers to head nodes of the trees are stored. Table 4.2 shows the number of segments (n_d) into which the contour has been divided, depth of the threaded binary tree, upper left corner of the MBR of contour(UL), width of MBR in x and y directions, starting and ending leaf nodes(LN) of each contour. The relational matrix will have non zero



(a) Contour Layer of Topomap 2 (b) Contour Line of Figure 4.7(a)

Figure 4.7: Representation of Contour Layer

Table 4.1: Representation of Contour Layers shown in Figure 3.22

Contour layer	No of cc's	No of segments/LN's	Max depth of tree
1	268	323	5
2	236	481	5
3	254	300	5
4	176	234	4
5	553	655	4
6	330	397	3
9	928	975	3
11	438	566	3
12	413	452	3

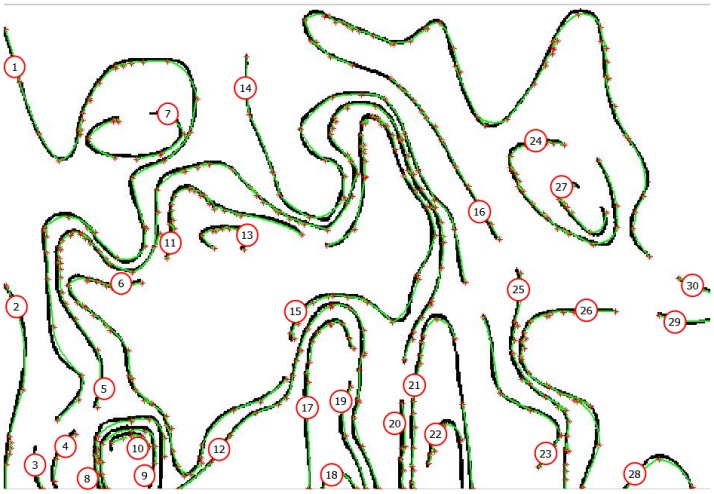


Figure 4.8: Representation of Contour Layer, shown in Figure 4.8

values at the positions where gaps have to be filled between the contours or with in a single contour. By backtracking the matrix, the final components will be

- (1,3)
- (4,5),(5,20)
- (6,11),(11,15),(15,6)
- (17,19)
- (14,23)
- (30,16),(16,25)
- (29,26)
- Closed contours are 7,9,10,13,22,24,27
- Contour segments whose end segments end at the boundaries of the topomap are 2,8,12,18,21,28

This representation can further be used to tag altitude values to the contour lines, which is discussed in subsequent chapters.

Table 4.2: Results of Contour Layer shown in Figure 4.8

Cont Seg	No of Seg	Depth of tree	UL of	MBR	Width of	MBR	Start LN	End LN
1	17	6	0.5	16.5	139	284	16	27
2	1	1	0.5	201.5	17	148	2	2
3	1	1	20.5	318.5	7	31	2	2
4	1	1	34.5	306.5	18	43	2	2
5	23	6	36.5	69.5	281	222	31	11
6	9	4	45.5	194.5	159	146	15	8
7	1	1	59.5	76.5	70	34	2	2
8	1	1	64.5	296.5	51	53	2	2
9	1	1	68.5	303.5	42	46	2	2
10	1	1	76.5	308.5	28	16	2	2
11	3	2	116.5	129.5	100	54	3	5
12	9	4	125.5	214.5	145	135	15	9
13	1	1	141.5	158.5	37	19	2	2
14	15	5	174.5	35.5	159	166	15	8
15	9	5	206.5	78.5	105	165	13	49
16	20	6	216.5	2.5	251	180	15	14
17	3	2	216.5	226.5	36	123	3	4
18	1	1	229.5	335.5	25	14	2	2
19	1	1	242.5	271.5	20	78	2	2
20	1	1	285.5	285.5	4	64	2	2
21	3	2	293.5	223.5	46	126	7	6
22	1	1	305.5	299.5	27	50	2	2
23	4	3	345.5	223.5	57	112	7	2
24	4	3	364.5	96.5	79	78	3	4
25	4	3	365.5	189.5	43	160	7	2
26	7	4	372.5	219.5	70	130	3	10
27	1	1	399.5	126.5	37	41	2	2
28	1	1	447.5	324.5	51	25	2	2
29	1	1	472.5	222.5	38	9	2	2
30	1	1	486.5	196.5	24	12	2	2

Chapter 5

Gap Identification and Pairing of Contour Segments

A binary contour layer should ideally consist of contour lines which are either closed loops or whose ends are on the edges of topomap. Figure 5.1(a) shows the topomap and 5.1(b) shows the clean contour layer extracted from it.

However, as can be seen in the Figure 5.1(b), the contour lines are broken, gaps of varying length formed as

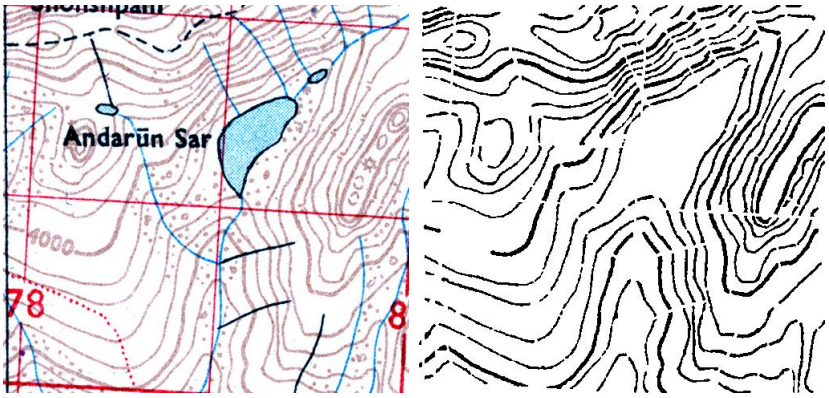
1. contour lines are always intersected or overlapped with other features, as seen in figure 5.1(a).

2. colours of the same contour line are not uniform due to aliased and false colours. Figure 5.2(a) shows the plot of RGB values of a single contour line.

3. topographic maps also use few supplementary contour lines, apart from continuous intermediate contour lines and index contour lines, which are discontinuous lines. An example is shown in figure 5.2(b) (CWQ06a).

The process of identifying and filling the gaps of binary contour layer is termed as Reconstruction. This chapter describes the algorithms employed for identifying the gaps in the contour layer which has been extracted from a topographic map. Automatic reconstruction of the contour layer is complex as the contour lines are

- Of varying width (Width varies within a single contour line, i.e they are not smooth and between different lines in a map).

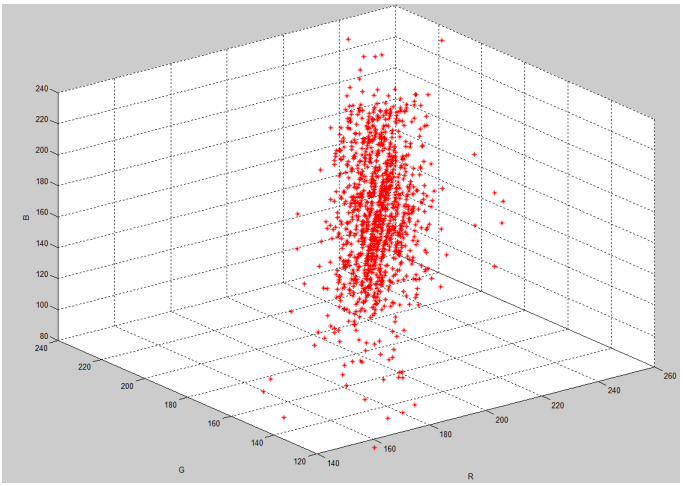


(a) Topographic map (b) Contour layer

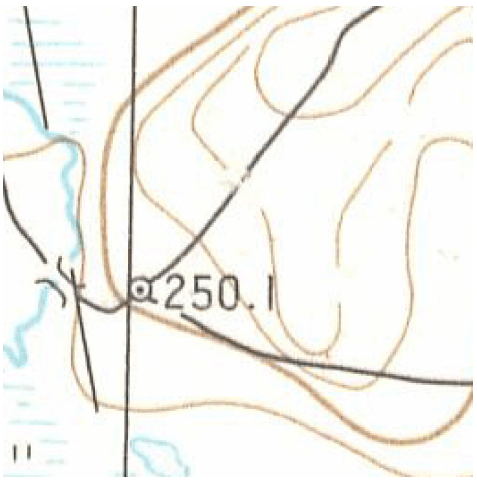
Figure 5.1: Contour Layer Extraction

- Of different shapes, contour lines can be in elliptical shape and concentric or can be more like curved lines parallel to each other.
- Closely spaced(ex: depicting hilly terrain), that it becomes difficult to identify the background between the lines.

Traditional methods including line-tracking method pay much attention to the image characteristics on the pixel scale and have difficulty in recognizing and connecting broken parts of contour lines (JY04). Therefore, identification and filling of the gaps is an essential module in automatic digitization of contour layer. Methods focussing on reconstruction of contour lines have been developed, which can be classified as either Image based / Geometric based, which use either Local/Global approach and which use external information or not. In the image based approach, two segments are grouped based on the perceptual principles like proximity and continuity. The basic limitation of Image based methods is the lack of utilization of domain knowledge e.g. contour lines do not intersect. Arrighi et al. (AS99) have used euclidean distances between extremities and weighted directions and combined both to compute a new distance for connecting the broken lines. Once all possible distances are computed, the extremities in lower distance are connected. The procedure is repeated until no extremities are left. Geometric based approaches, on the other hand, transform the problem of raster-to-vector conversion into curve reconstruction. Spinello Salvatore et



(a) RGB values of Contour line



(b) Intermediate Contour lines

Figure 5.2: Contour Line Properties

al. (SG04) have vectorized contour line using Delaunay Triangulation where Delaunay edges are filtered using local and global rules. Du Jinyang et al. (JY04) have used principles of mathematical morphology (dilation) to acquire spatial relationships of contour lines for matching and connecting broken contour lines. Lastly, gradient vector flow approaches employ orientation field generated by the contour lines for reconstruction. An application to snakes is given in (XZZ06).

Most of the methods mentioned above rely on the availability of the end points of the contour lines for subsequent pairing and gap filling. The methods essentially differ in the way the end points are paired by using perceptual principles or curve reconstruction, based on local or global information. In this work, the focus has been on the detection of end points of contour line segments which contribute to gaps, consequently facilitating gap detection. The proposed approach leads to a significant reduction in the complexity in the process of pairing contour line segments.

Main steps to be followed in reconstruction are

1. Gap identification
2. Pairing/Matching Endpoints
3. Gap Filling

The first step to reconstruct the contour layer is to identify the broken ends of the contour lines. The Gap Identification process primarily detects broken ends of the contours. Every broken end of the contour line should be joined to another broken end or should be routed to edge of the image. Other characteristic of contour lines is that contour lines never intersect each other. This property has to be maintained during the matching process. Once the ends of contour segments are matched, the gap has to be filled between the contour lines, i.e. the pixels of the background which belong to the contour should be identified.

Section 5.1 describes the algorithms for isolating the segments of contours which contribute to a gap and identifying their end points. Section 5.2 describes the parallel implementation of the Gap Identification algorithm. Subsequent to Gap identification, a challenging problem of pairing gap forming contour segments

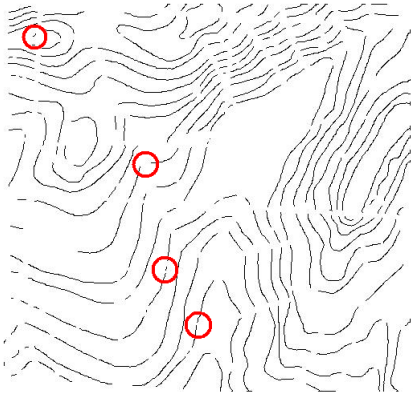


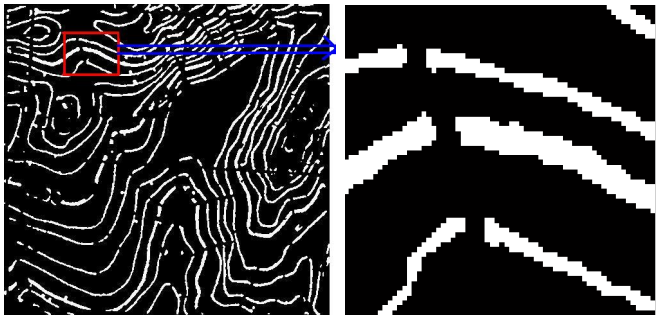
Figure 5.3: Thinning of Contour Layer

arises, which is discussed in Section 5.3. Results and Discussion are placed in Section 5.4.

5.1 Gap Identification

The classical techniques usually first skeletonize the lines, and then seek for relevant extremities of the skeleton to recover the missing parts of the lines. However, problem occurs because skeletonization process is very sensitive to noise and creates small branches at the edge of the lines. Although these can be removed by pruning, the lines become shorter; therefore some relevant information will be lost (SYSbNI04). For example, the Figure 5.3 shows the result of applying Zhang's thinning algorithm (TC84) on a contour layer. As it can be seen that thinning introduces more problems by increasing the gap width and in some cases introducing new gaps. Hence we have approached the problem of finding broken ends of contours without thinning the contour lines.

The Gap Identification operates by reducing the binary contour layer to one consisting of segments of those contour lines which contribute to gaps. The resulting image is defined as gap contour image. An additional feature of our algorithm is finding the end points of the segments. The end points play an important role in matching of contour segments and gap filling. The detailed description of steps in the algorithm are given in the following sections.



(a) Contour Layer (b) Cropped Image

Figure 5.4: Portion of Contour Layer

5.1.1 Gap Segment Isolation

The first step in Gap Identification is isolating the contour segments which are associated with gaps which is accomplished by making use of a sliding window. The binary image obtained from the extraction process (Ref: Img_{f2} from Algorithm 3.6) is padded with rows or columns of background pixels so that the image is square and is in the power of two. Let this be denoted by Img_s. The steps are explained using a portion of contour layer shown in Figure 5.4(b), which is of size 64×64

Extraction of Contour Segments

i) A $w_g \times w_g$ window is slid across the image with a horizontal or vertical overlap of $w_g/2$ between successive positions of the window. If the image size is $N_{img} \times N_{img}$, where $N_{img} = 2^p,\ w_g = 2^q,\ \forall q \leq p$, number of windows/sub images are $9^{log_2 N_{img}/w_g}$. Let this be denoted by N_w. For example if $N_{img} = 64$ and $w_g = 32$, nine windows are formed as shown in the Figure 5.5. For the figure shown in 5.4(b) nine sub images each of size 32×32 would be as shown in the Figure 5.6.

ii) In each window, contour segments are found using connected component analysis (CCA). Figure 5.7 shows the CC of the first subimage.

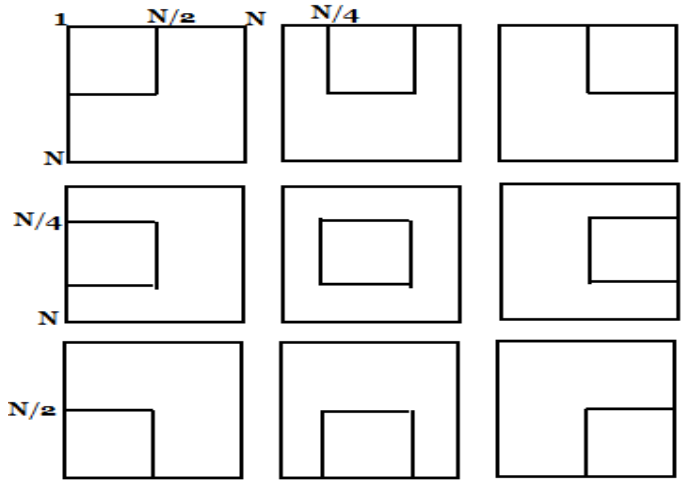


Figure 5.5: Sliding Window

Detection of Segment End Points

The end points of the contour segments extracted from the previous step need to be found out. This is achieved as follows:

i) Fit a Minimum Bounding Rectangle to the contour segment.

ii) Find the intersection of curve with the edges of the rectangle. There will be many intersecting points along each edge of the MBR due to thickness of the contour segment. However, we consider only one representative point from each set of connected points along the edge.

iii) A pixel is a candidate for being an end point only if it has at least one entire row and one entire column of background pixels in its 5×5 neighbourhood.

The Figure 5.8 shows the end points identified on a contour segment of the first subimage using green *.

Isolation of Segments causing Gaps

The following assumptions are in place:

i) For any contour segment causing a gap, there exists at least one window wherein the end point of the segment does not lie on the boundary of the window.

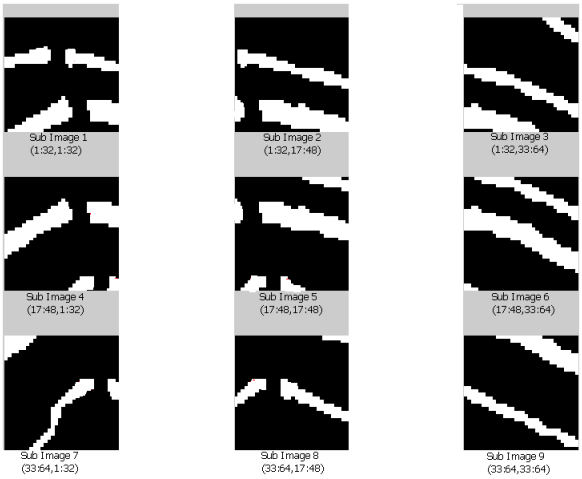


Figure 5.6: Sub Images

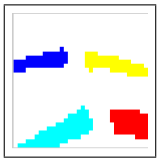


Figure 5.7: CC of Sub Image 1

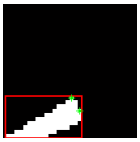


Figure 5.8: End point Detection

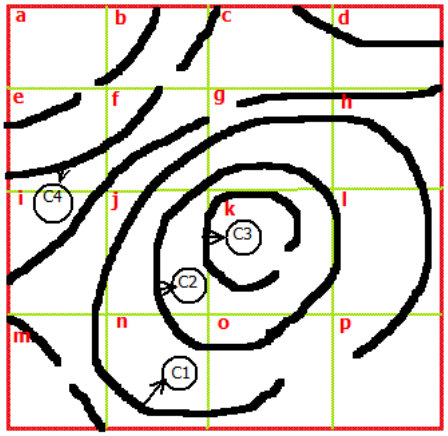


Figure 5.9: Portion of Contour Layer

ii) On the other hand in any given window if the contour segment in the window has both its end points on the boundary, then the portion of the contour within that window does not contribute to the gap.

iii) If at least two of the end points of a contour segment lie on the window boundary, then, within the window the contour segment does not contribute to a gap.

iv) A contour which has a gap should have at least one end point which does not lie on the edge of the window.

The assumptions are illustrated with the help of Figure 5.9. The green lines in the Figure indicate the windows into which the contours have been divided, i.e. a - p. The overlapping windows have not been shown for clarity. It can be seen that the contour C1 exists in the windows f, g, h, j, l, m, n, o, p. The segments of C1 in all the windows except in windows o and p have end points on the boundaries of the respective windows. Hence according to assumptions ii) and iii) those segments of C1 do not contribute to a gap. Segments of C1 in windows o and p are gap segments according to assumption iv) and hence contour C1 has a gap. In the case of contour C2, the segments end at the boundaries in all the windows and hence is not a gap contour. It is to be noted that if the windows are divided as shown in the figure without overlap, contours C3 and C4 will not be detected as gap contours according to the assumptions made above. Hence it is essential to

use overlapping windows as this would ensure the detection of varying lengths of gaps formed between different shapes of contours.

The end points returned from the previous step are tested to check as to how many of them are on the boundary of window i.e. if the row or column is equal to C. If a segment in a window consists of an end point that does not line on the boundary of the window (i.e. C), it is classified as a gap segment. Using these criteria, we isolate contours contributing to gaps in each window and label them. A labelled image storing only the gap segments is formed, (GI) from each sub-image.

Let the number of windows with gap segments be $N_g, (\leq N_w)$ and the total number of gap segments be T_g. A vector V_g stores the number of gap segments in each window.

$$\sum_{i=1}^{N_g} V_g(i) = T_g$$

Let W denote the set of labelled images, $\{GI_s, s = 1\ to\ N_g\}$ where, $max(GI_s) = V_g(s)$. For the image shown in Figure 5.6, of the 9 sub images, six images have segments of contours contributing to gaps. Hence $N_g = 6, T_g = 20$ and V_g =(4, 3, 4, 4, 2, 3). This local information is combined to get the gap segments of the whole image, as described in the next section.

The main steps of the algorithm are presented in the Procedure 3.

Procedure 3 GAPSEGMENTS(Img_w)

//Find segments of contours which form a gap//
Input:
Img_w: Portion of contour image in a window
Output:
GI_s: Set of images which store gap segments
$Epts$: End points of gap segments
Var:
N_{gap}: Number of gaps
Method:
1: Find the Connected components of Img_w
2: $N_{gap} \leftarrow 0$
3: **for** each component **do**
4: Find end points of segment(CC)
//Formation of Gap segment image//

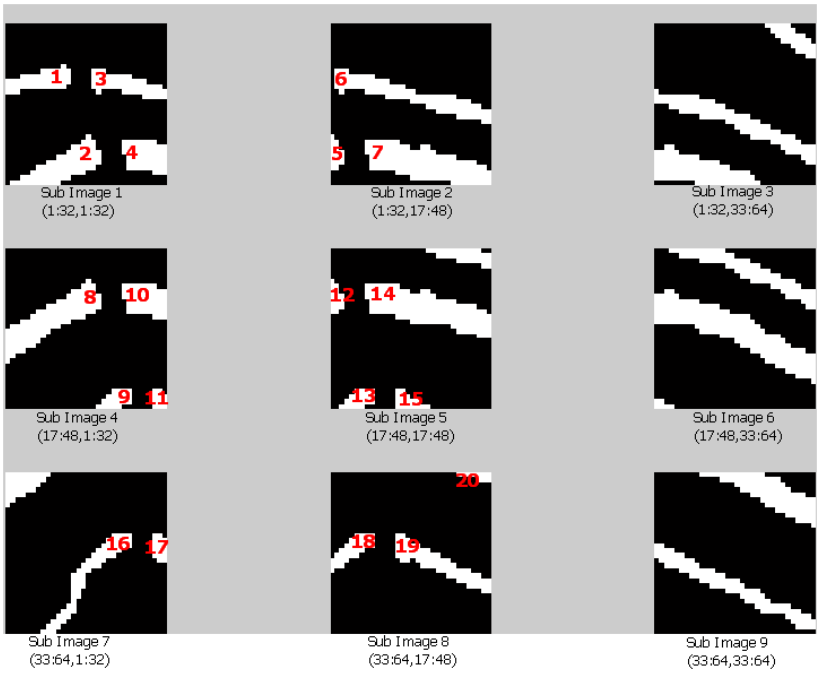


Figure 5.10: Sub Images with identified Gap segments, labelled

5: **if** end points not on boundary of window **then**
6: $N_{gap} \leftarrow N_{gap} + 1$
7: Find location of segment pixels in the image Img_w and update image GI_s by storing N_{gap} in those positions
8: Add end points of gap segments to $Epts$
9: **end if**
10: **end for**
11: **return** GI_s, $Epts$
end GAPSEGMENTS

5.1.2 Generation of Gap Contour Image

Consequent to the identification of the broken ends of segments in each sub image, all the segments which belong to a single contour should be merged. The gap segments identified in the sub images shown in Figure 5.6, are labelled as shown in Figure 5.10. This local information should be combined, for example in the first two sub images, gap segments 2 & 5, 3& 6, 4& 7 should be combined. This process of collapsing the segments is explained below:

i) Let S denote the set of binary images $\{I_g,\ g = 1\ to\ T_g\}$, each image representing one gap segment.

ii) The information of gap segments of all the windows is combined by preparing the Adjacency matrix, A of size $T_g \times T_g$.

The value at the position (i,j) and (j,i) in matrix A is 1 if $I_i \cap I_j \neq 0$, $where\ I_i \& I_j \in S$. It is to be noted that since overlapping window has been used, if two segments in windows belong to the same contour, they share common pixels and hence their intersection will not be null. An adjacency matrix having "1" entries for components having common pixels is prepared.

For example for the images shown in Figure 5.6 an adjacency matrix of size 20×20 is formed. A matrix formed between the first two sub images is shown below

	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0
2	0	1	0	0	1	0	0
3	0	0	1	0	0	1	0
4	0	0	0	1	0	0	1
5	0	1	0	0	1	0	0
6	0	0	1	0	0	1	0
7	0	0	0	1	0	0	1

iii) Backtracking algorithm based on graph theory is used to collapse the segments from the adjacency matrix. Let N_c be the number of components and V_c be the the vector of size T_g storing to which component, the gap segment belongs to. Hence T_g segments are divided into N_c components. For the matrix generated on the image given in the Figure 5.6, 20 segments are divided into six components as, $\{\{1\}, \{2, 5, 8, 12\}, \{3, 6\}, \{4, 7, 10, 14, 20\}$,

Figure 5.11: GC Image

$\{9, 13, 16, 18\}, \{11, 15, 17, 19\}\}$. It can be observed from the figure that the image has 3 gaps and 6 contour segments contribute to the gaps, and hence 6 components are obtained.

iv) An image having only those contours contributing to gaps is formed which is henceforth named as the Gap-Contour(GC) image. GC is a labelled image, each image pixel having a value from 1 *to* N_c. The GC image consists of clusters of contours that cause gaps along with the corresponding end points. In general, while operating on the entire topomap, the number of clusters will be twice the number of gaps. In specific cases - where the gap formed is between a contour and the boundary of the topomap, number of clusters would be less. Figure 5.11 shows the GC image of image shown in Figure 5.4.

We have developed a recursive version of Gap Identification algorithm which has been presented in Algorithm 5.1. Results on portions of contour layer are shown in Figure 5.12.

5.2 Parallel Implementation

5.2.1 Introduction

The task of processing topomaps is both data intensive and computing intensive. So implementation of these algorithms on a parallel architecture is a viable solution as this

Algorithm 5.1 GAPIDEN(Img_s,N_img,N,$strow$,$stcol$)

//Recursive Gap Identification algorithm //

Input:

Img_s:	Input contour image(binary)
N_img:	The size of input image
N :	Size of sub images
$strow$, $stcol$:	Starting row,column of window in Img_s

Output:

$GCIm$:	Gap Contour Image
$Epts$:	End points of gap segments

Var:

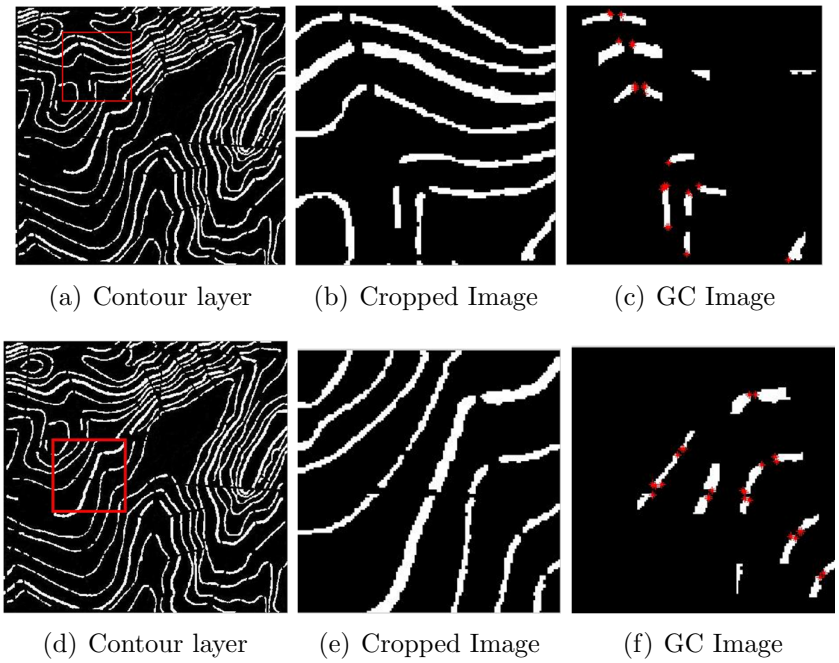
w_g:	Size of window (Ref: Section 5.1.1, Step i)
N_g:	Number of windows consisting of gap segments
$GCTmp$:	Gap Contour image in a window
W:	Set of Gap contour images initialized to null
V_g:	Vector storing number of gap segments in each window
A_g:	Adjacency Matrix, of size $[Sum(V_g), Sum(V_g)]$
i, j:	Loop indices

Method:

1: **if** $N_{img} = w_g$ **then**
2: Call GAPSEGMENTS with the image in window between rows (strow, strow+N-1) and between columns (stcol, stcol+N-1). The procedure returns GI_s, $Epts$
3: **return**
4: **end if**
5: $N_g \leftarrow 0$
6: **for** $i \leftarrow 1$ *to* 3 **do**
7: $col \leftarrow stcol$;
8: **for** $j \leftarrow 1$ *to* 3 **do**
9: **Call GAPIDEN(**Img_s**,**N_img**,**$N/2$**,**$strow$**,**col**)** //Returns $GCTmp$ and $Eptstmp$//
10: **if** $GCTmp \neq$ NULL **then**
11: $N_g \leftarrow N_g + 1$
12: $V_g(N_g) \leftarrow max(GCTmp)$
13: $W \leftarrow W \cup GCTmp$
14: **end if**
15: $col \leftarrow col + N/4$
16: **end for**
17: $strow \leftarrow strow + N/4$
18: **end for**
19: **if** W is not null **then**
20: Compute Adjacency Matrix, A_g from set of images in W
21: Compute components of A_g using backtracking algorithm
22: Form $GCIm$ and $Epts$ from the components
23: **end if**

end GAPIDEN

Computational Complexity of GAPIDEN
Recurrence relation :
$T(n) \quad = \quad N_w \times O(n^2)$
where n = number of pixels in each segment



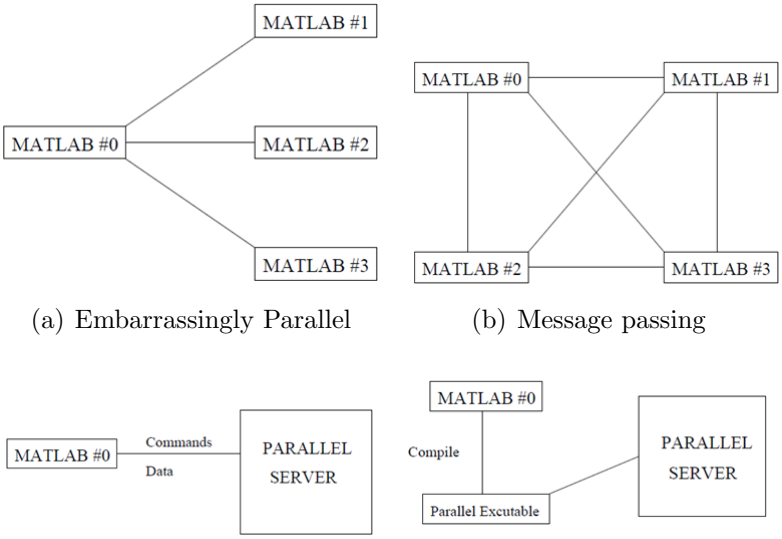
(a) Contour layer (b) Cropped Image (c) GC Image

(d) Contour layer (e) Cropped Image (f) GC Image

Figure 5.12: Gap Identification

1. Enables processing of high dimensional data.
2. Meets the computational requirements for complex analysis of images.
3. Provides faster response time.
4. Caters to the inherently data parallel organization of images (the input image data required to compute a given portion of the output is spatially localized).

Software development tools that are specifically tailored to image processing applications have been developed to



(a) Embarrassingly Parallel (b) Message passing

(c) Backend Support (d) Matlab Compilers

Figure 5.13: Parallel tools for Matlab usage

1. alleviate the problem of low-level software design for parallel and distributed computers.

2. provide higher-level abstractions to the user than general purpose tools

3. incorporate important domain-specific assumptions

Ex: PIPT - Parallel Image processing toolkit based on message-passing model of parallelism (JAR98).

Matlab has been used to implement all the algorithms as it is one of the most commonly used languages for scientific computing. Hence to test the efficiency of paralleling the algorithms implemented in Matlab, a survey has been done for tools/mechanisms which enable to run these Matlab programs on parallel architectures. Parallel MATLAB has been an active area of research for a number of years and many different approaches have been developed. Table 5.1 shows the list of tools under various parallel architectures(shown in Figure 5.13), which have been developed to use along with Matlab (CEO05).

Table 5.1: Parallel Tools for Matlab Usage

Type	Features	Examples
Embarrassingly Parallel	• Makes use of multiple MATLAB processes running on different machines or a single machine with multiple processors. • No coordination between the MATLAB processes is provided. • A parent process passes off data to the child processes, which work on local data, and return the result to the parent process.	Multi, Paralize, PLab, Parmatlab
Message Passing	• Provides message passing routines between MATLAB processes • Enable users to write their own parallel programs in MATLAB in a fashion similar to writing parallel programs with a compiled language using MPI. • Advantage of flexibility	MultiMATLAB, CMTM, DP-Toolbox, MPITB, MATmarks, PMI, PTToolbox, MatlabMPI, pMatlab
Backend Support	• Uses MATLAB as a front end for a parallel computation engine, which usually makes use of high performance numerical computation libraries like ScaLAPACK • Advantage is that it requires one MATLAB session • Usually does not require the end user to have knowledge of parallel programming.	NetSolve, DLab, Matpar, PLAPACK, PARAMAT , MATLAB*P
Matlab Compilers	• Software's compile MATLAB scripts into an executable or sometimes translates them into a compiled language as an intermediate step. • Advantage that the compiled code runs without the overhead incurred by MATLAB and hence MATLAB is used as a development platform instead of a computing environment. • Allows the produced parallel program to run on platforms which does not support MATLAB (e.g. SGI).	Otter, RTExpress, ParAL, FALCON, CONLAB, MATCH, Menhir

5.2.2 Implementation Using pMatlab

The pMatlab library is designed and implemented at MIT Lincoln Laboratory (BKKR07). It uses MatlabMPI (JS04), which is a pure MATLAB implementation of the most basic MPI functions as the communication layer. Figure 5.14 illustrates the layered architecture of the parallel library (BKKR07). In the architecture, the pMatlab library implements distributed array constructs.

pMatlab has been used as it

1. Provides high level parallel data structures and functions.
2. Parallel functionality can be added to existing serial programs with minor modifications
3. Pure Matlab implementation
4. Distributed matrices/vectors are created by using maps that describe data distribution. Data overlap, required for some image processing applications, is also supported through the map interface.
5. Offers subset of MPI functions using standard Matlab file I/O. The functions used by pMatlab are listed in Table 5.2
6. Publicly available: http://www.ll.mit.edu/MatlabMPI & http://www.ll.mit.edu/mission/isr/pmatlab/pmatlab.html

pMatlab distributes data among the processors using a new data type dmat - distributed matrices. How and where the dmat must be distributed is accomplished with a map object. To create a dmat, the programmer first creates a map object, then calls one of several constructor functions, passing in the dmats dimensions and the map object. pMatlab includes various functions that specifically address the global and local scopes of dmats. Each map object has four components:

1. The grid is a vector of integers that specifies how each dimension of a dmat is broken up. For example, if the grid is [2 3], the first dimension is broken up between 2 processors and the second dimension is broken up between 3 processors, as shown in Figure 5.15(a)

Table 5.2: MPI functions provided by MatlabMPI

Function Name	Function Description
MPI_Init	Initializes MPI
MPI_Comm_size	Gets the number of processors in a communication
MPI_Comm_rank	Gets the rank of current processor within a communicator
MPI_Send	Sends a message to a processor
MPI_Recv	Receives a message from a processor
MPI_Finalize	Finalizes MPI

2. The distribution specifies how to distribute each dimension of the dmat among processors. There are three types of distributions:

   - Block: Each processor contains a single contiguous block of data
   - Cyclic: Data are interleaved among processors
   - Block-cyclic: Contiguous blocks of data are interleaved among processors.

   Figure 5.15(b) shows an example of the same dmat distributed over four processors using each of the three types of data distributions.

3. The processor list specifies on which ranks the object should be distributed. Ranks are assigned column-wise (top-down, then left-right) to grid locations in sequential order (Figure 5.15(c)).

4. Overlap is a vector of integers that specify the amount of data overlap between processors for each dimension. Only block distributions can have overlap. Overlap is useful in situations when data on the boundary between two processors are required by both, e.g. convolution. In these cases, using overlap copies boundary data onto both processors. Figure 5.15(d) shows an example of a dmat distributed column-wise across four processors with 1 column of overlap between adjacent processors.

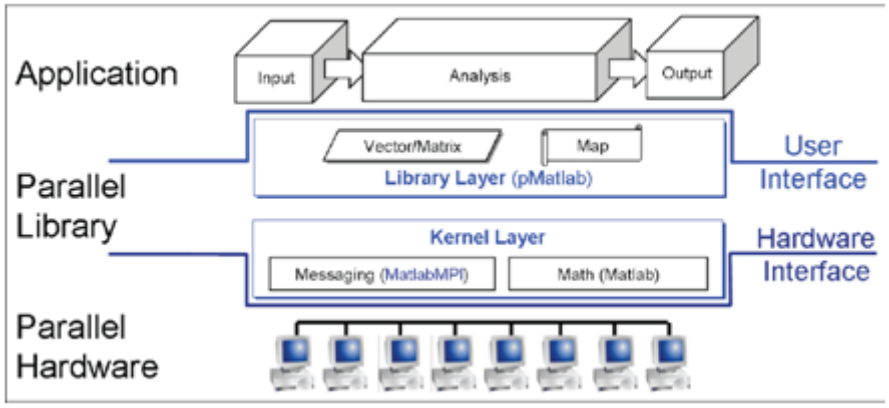
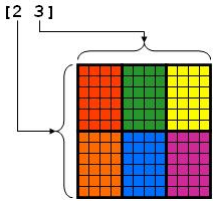
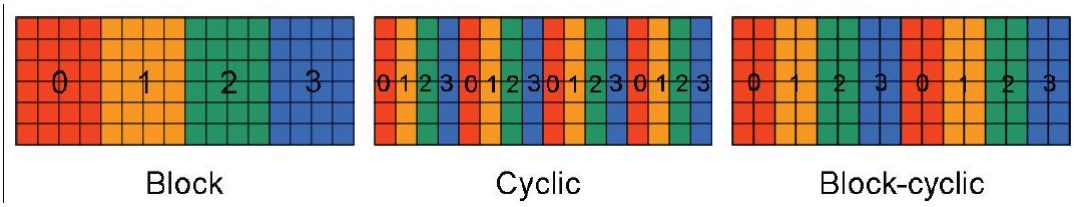


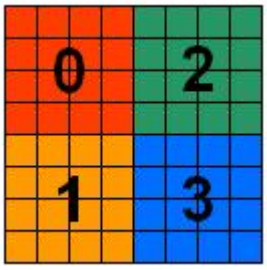
Figure 5.14: pMatlab Architecture (BKKR07)



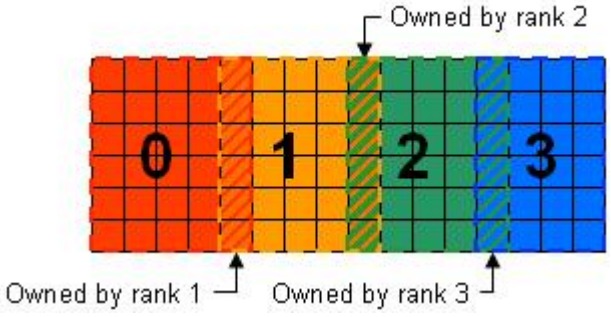
(a) Map Grid



(b) Data distribution



(c) Processor assignment to grid locations



(d) data overlap between processors

Figure 5.15: Data distribution using Map Object

5.2.3 Parallel Gap Identification Algorithm

The Gap Identification algorithm, described in the previous section is amenable to parallel implementation as the algorithm can be executed on different parts of the topomap and the results of each part of the image can then be collated. The gap identification algorithm described in the previous section is run by each process on a portion of contour map. Algorithm 5.2 shows the details of how Gap Identification can be run using P-RAM model of computation (GR88).

Algorithm 5.2 PAR_GAPIDEN

//Parallel Gap Identification Algorithm//
Input:
Img_s: Binary Contour image
n_p: Number of processors
Output:
$GCIm$: Gap Contour Image
$Epts$: End points
Method:
Divide Img_s into n_p partitions
for all $p,\ 1 \leq p \leq n$ in parallel **do**
Call GAPIDEN with Img_s^p
Store return values $GCIm^p,\ Epts^p$
end for
Combine results from n_p to obtain $GCIm,\ Epts$

Computational Complexity of PAR_GAPIDEN
$T(n) = \frac{N_w \times O(n^2)}{n_p}$
using n_p processors
where n = number of pixels in each segment

The contour map is distributed using the map object of pMatlab library which has been described in the previous section. For example if the contour map is of size 512 × 512, the image can be divided into 4 images of size 256 × 256 and each run on a different processor. Map object will then be defined as, map([2 2],{ },0:3). Pseudo code of the algorithm is given in Procedure 4.

Procedure 4 pMatlab_GI(I_{inp})

// Parallel Implementation of GI using pMatlab //

Method:

1: pMatlab_Init
2: pMATLAB.comm_size
3: my_rank ← pMATLAB.my_rank
   //Create Maps//
4: mapImage ← map([4 4],,0:Ncpus-1)
5: mapedpts ← map([Ncpus 1],,0:Ncpus-1)
   // Allocate data//
6: (n_x n_y) *gets* size(inpimg)
7: imOv ← zeros(n_x,n_y,mapImage)
8: outim ← zeros(n_x,n_y,mapImage)
9: glendpt ← zeros(n_x,2,mapedpts)
   //Get the local data of each processor//
10: (myI myJ) ← global_ind(imOv)
   //Returns the indices owned by the local processor//
11: imOv_local ← local(imOv)
   //Returns a matrix that contains the section of imOv that resides on the local processor//
12: imOv_local ← inpimg(myI(1) : myI(end),myJ(1) : myJ(end))
13: imOv ← put_local(imOv,imOv_local)
   //Writes the contents of a matrix into that processors section of imOv//
   //Call Gap Identification on local data//
14: (lclus lendpts) ← GI(imOv_local,length(myI),length(myI),1,1)
   //Put the local results of each processor//
15: outim ← put_local(outim,lclus)
16: glendpt ← put_local(glendpt,lendpts)
   //Combine the results//
17: clus ← agg(outim)
   //Aggregates the parts of outim into a whole and returns it as a regular double matrix, clus//

Table 5.3: Execution of Parallel Gap Identification

Image size	Execution Time(in secs)		
	np = 1	np = 4	np = 16
128 X 128	4	3.71	4.86
256 X 256	39	12	5.84
512 X 512	-	90	11.72

18: endpts ← agg(glendpt)
//Finalize the pMATLAB program//
19: pMatlab_Finalize
end pMatlab_GI

The Gap identification algorithm has been implemented using pMatlab library. Data (Contour map) is distributed among the processes and the algorithm is run by each process on its local part of the data. Table 5.3 shows the execution times for various sizes of images when applied on the contour layer given in Figure 5.12(a) i.e. contour layer extracted from Topomap 1 shown in Figure 3.22 in Chapter 3 .

5.3 Pairing Gap Segments

Once the contour segments in the vicinity of a gap are isolated, pairs of contour segments which are supposed to close the gap are found by matching. A brute force matching algorithm would consider pairing each contour segment with every other contour segment in the neighbourhood of the gap compared with rest of the contours. A gap can be attributed to two end points. It is worth mentioning that all the pairs of end points need not be meaningful. Hence as pairing endpoints for gap filling is free from global configuration, algorithms for identifying the possible regions for paring of contour end points have been discussed in Section 5.3.1.

5.3.1 Localization

To reduce the complexity of pairing it is required first to identify potential end points of the segments contributing to a given gap. This is possible by classifying

pixels in the gap and surrounded by the gap either to the contour or background i.e. the considered pixels should be clustered to two clusters, one belonging to the contour and other to the background. This clustering problem (2 clusters) has been attempted by one of the popular algorithms by name EM algorithm.

Expectation maximization algorithm(EM)

The EM algorithm is an iterative algorithm for calculating the maximum-likelihood or maximum-a-posteriori estimates of finite mixture models, when the observations can be viewed as incomplete data i.e. when there is a many to one mapping from underlying distribution to the distribution governing the observation (ALR77). Each iteration of the algorithm consists of an expectation step (E-Step) followed by a maximization step (M-Step). The E-Step is with respect to the unknown underlying variables, using the current estimate of parameters and conditioned upon the observation, the M-Step then provides new ML estimates/updates of the parameters given in the E-Step. These two steps are iterated until convergence. To use EM Algorithm, the number of component densities 'C' in the mixture has to be known and an initial guess for the value of component parameters $\Theta^{(0)} = (\mu_k^{(0)}, \sigma_k^{(0)})$ has to be made. Once initial estimates are found, the parameter estimates are updated using iterative EM update equations. Details of EM for Gaussian Mixture models (GMM) are given below:

// EM algorithm for Gaussian Mixture Models (ALR77) //

Step 1: Determine the number of Components 'C in the mixture.

Step 2: Determine the Initial Guess for parameters $\theta^{(0)}$, the component parameters. These are the Mixing Coefficients(π) , Mean (μ), Standard Deviation (σ) , for each normal density. $t = 0$; $\Theta^{(0)} = (\pi,\ \mu,\ \sigma^2)$

Step 3: E-Step: $t = t + 1$

For each data point X_i, determine the posterior probability by applying Bayes Rule.

$$\gamma_{ik} = \frac{\hat{\pi}_k f(X_i; \hat{\mu}_k, \hat{\sigma}_k)}{F(X_i)} \quad i = 1, 2, 3.., N; k = 1, 2, .., C$$

where γ_{ik} represents the estimated posterior probability that point X_i belongs to the k-th component density, and $f(X_i; \hat{\mu}_k, \hat{\sigma}_k)$ is univariate normal density for the k-th component density given by

$$f(X_i; \hat{\mu}_k, \hat{\sigma}_k) = \frac{1}{\hat{\sigma}_k \sqrt{2\pi}} e^{\frac{-(X_i - \hat{\mu}_k)^2}{2\hat{\sigma}_k^2}}$$

and Finite mixture estimate at the point X_i :

$$F(X_i) = \sum_{k=1}^{C} \hat{\pi}_k f(X_i; \hat{\mu}_k, \hat{\sigma}_k)$$

The posterior probability tells us the likelihood that a point belongs to each of the separate component densities. This estimated posterior probability γ_{ik} is used to find the weighted updates of the parameters for each component.
Step 4: M-Step: Compute the weighted iterative ML estimates using posterior probability γ_{ik} obtained in Step 3 for each Component Density C_k

- Mixing Coefficients

$$\hat{\pi}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik}}{N}$$

- Mean

$$\hat{\mu}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik} X_i}{N}$$

- Standard Deviation

$$\hat{\sigma}_k^2 = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik} (X_i - \hat{\mu}_k)^2}{N}$$

$$\Theta^{(t)} = (\hat{\pi},\ \hat{\mu},\ \hat{\sigma}^2)$$

Step 5: If $\|\Theta^t - \Theta^{t-1}\| \geq \varepsilon$ then Repeat Steps 3 and 4 else return $\gamma,\ \Theta^t$

Details of how EM algorithm is used for pairing are given in Algorithm 5.3.

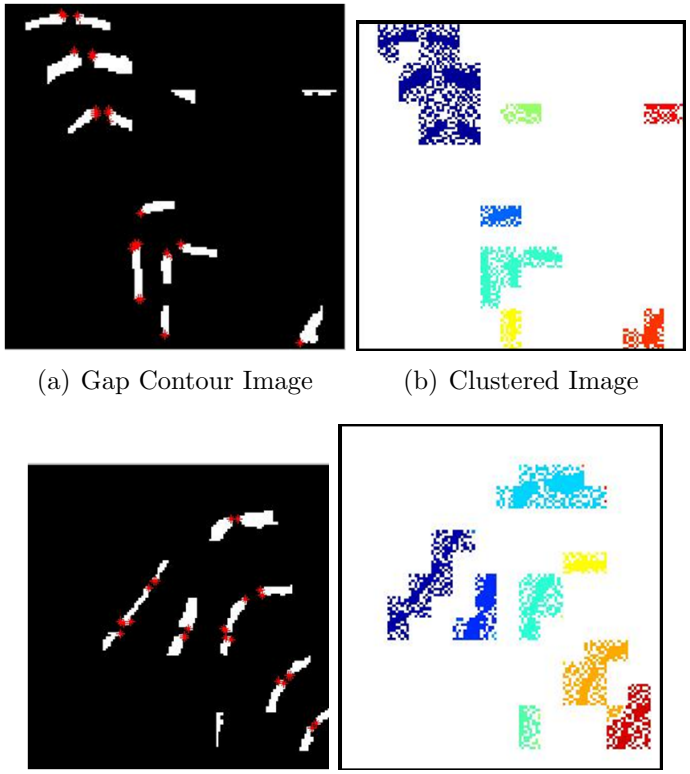
Figure 5.16 shows the results of applying Algorithm 5.3 on the Gap Contour Images, obtained from from Algorithm 5.1, shown in Figure 5.12 . The clouds

Algorithm 5.3 LOCALIZATION(GCI_m)

//Localize Pairing//
Input:
GCI_m: Gap Contour Image of size $[N_m, N_n]$(From Algorithm 5.1)
Output:
G Clouded Image of size $[N_m, N_n]$
Var:
D Vector of size $N_m \times N_n$ Method:

1: The Gap Contour image obtained from Gap Identification, GCI_m is converted to Gray scale from binary by saving it in the jpeg format. The gray level intensities of pixels are saved in vector D
2: The vector D is divided into 2 equal parts. μ_1, μ_2, σ_1, σ_2 are initialized to mean and standard deviation of each of these groups respectively
3: With Number of components equal to 2, (π) as 1/2 i.e. [0.5 0.5], μ_1, μ_2, σ_1, σ_2 as initial assignments for the parameters, D is submitted to EM algorithm
4: The posterior probabilities returned from EM procedure are used to classify the pixels into one of the components. Based on the mean and the standard deviations the components are labelled as belonging to background or object classes Detect the cloud formed by pixels identified as object pixels around gaps.
5: A connected component of this clouded image with the associated contour line segments are grouped, to obtain G. //Pairing problem is localized to this set//
6: **return** G

end LOCALIZATION



(a) Gap Contour Image (b) Clustered Image

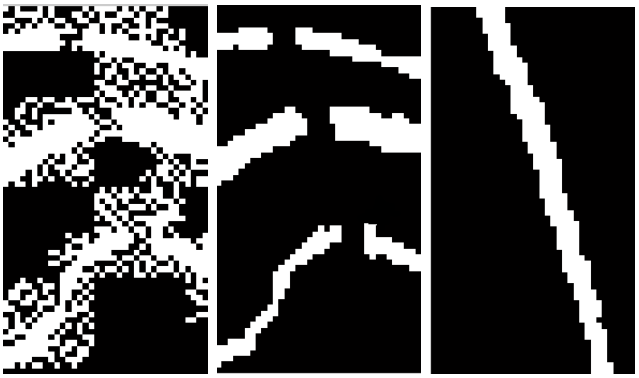
(c) Gap Contour Image (d) Clustered Image

Figure 5.16: Results of applying EM

are formed around contour segments. Each distinct colour indicates one localized set of contour segments to be subjected to pairing. For example, the cloud and segments in blue indicate the set of segments to be paired to fill the three gaps.

5.3.2 Local Pairing

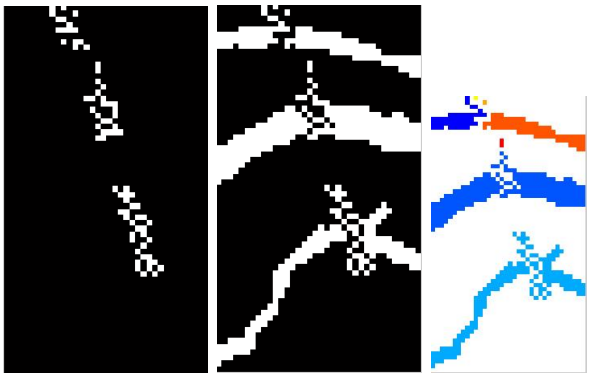
A set of contours are grouped using Localization, described in the previous section. The pair of contour segments of this set have to be matched. It has been discussed in the introduction of this chapter that gaps are mostly formed due to other intersecting or overlapping layers of topomap. Hence the pixels which belong to such a gap in the contour layer belong to any of the objects in the other layers. The cloud, formed as a result of applying EM, can be used to find out which pixels of other layers should belong to contour. This knowledge can be utilized for matching and gap filling of contours. Gap filling is discussed in the



(a) Component of clouded GC Image

(b) Contour Image

(c) Other layer Image



(d) Pixels of cloud and other layer

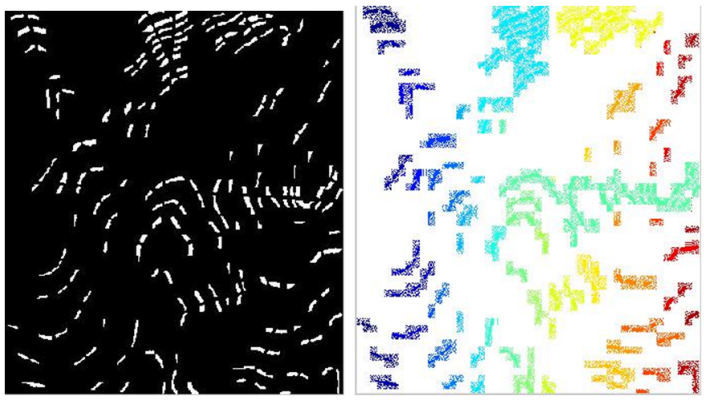
(e) Result Image

(f) CC of Result image

Figure 5.17: Local Pairing

next chapter, whereas matching is presented here. It is to be noted that as all the images are binary, logical operations can be formed between them.

1. Let the contour layer be C_I, other layer image be O_I, a component of clouded contour image be L_I. A component of clouded contour image shown in Figure 5.16(b) is shown in Figure 5.17(a), its corresponding contour layer is shown in Figure 5.17(b) and other layer is shown in 5.17(c). From the extraction stage, when contour layer is filtered from topomap, pixels of all other layers are grouped to obtain 'other layer'



(a) Gap Contour Image (b) Image after Localization

Figure 5.18: Results on Contour Layer of Topomap 1

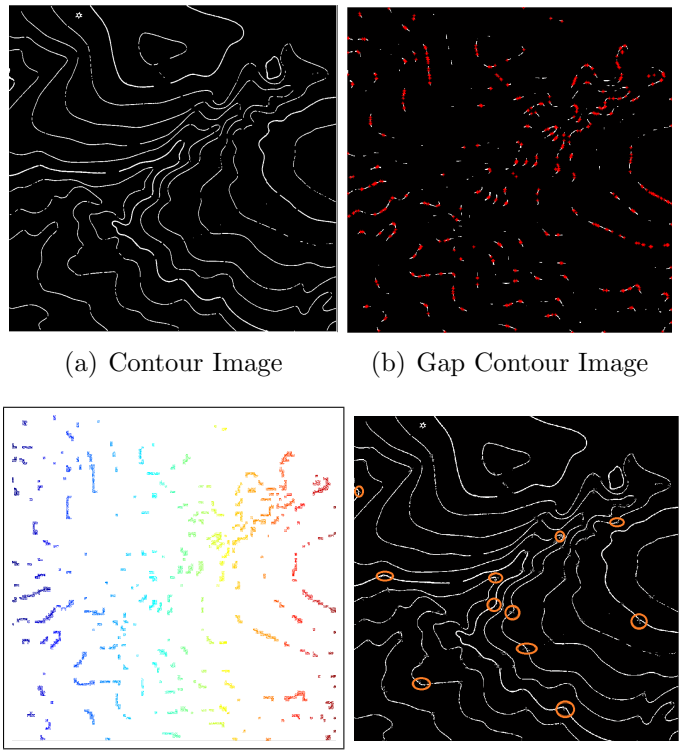
2. As some pixels in background in the contour layer should belong to the contour to fill the gap, some object pixels in the other layer should belong to the gap. An image is obtained by 'AND'ing O_I and L_I, which has pixels of cloud which also belong to the other layers. Let this image be A_I, shown in Figure 5.17(d).

3. A_I gives the pixels which belong to the other layer and to the gap in contour layer. Hence when A_I is 'OR'ed with C_I, an image which fills few gap pixels thereby connecting some contours is obtained, shown in Figure 5.17(e).

4. Connected component of image obtained in previous step gives a set of contours which are matched. The rest of the contours are evaluated for pairing by maintaining contour line properties which directs which pairs of end points need to be connected. the Figure 5.17(f) shows that among three sets of contours two sets have been paired. So the remaining one set of contours form a pair.

5.4 Results and Discussion

The Figure 5.18 shows the results of Gap Identification and Localization algorithm on the test image shown in Figure 5.1.



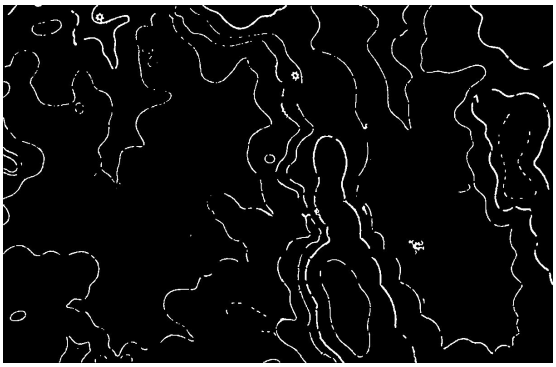
(a) Contour Image (b) Gap Contour Image

(c) Image after Localization (d) Image after Local pairing

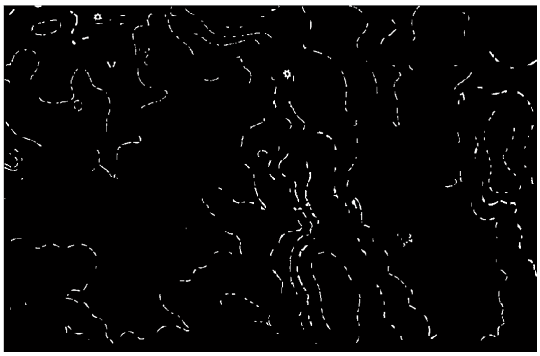
Figure 5.19: Results on Contour Layer of Topomap 2

Figure 5.19 shows the results on contour layer of topomap 2 shown in Figure 3.5. The Gap Identification algorithm is executed on portions of the contour layer and the results have been merged. The Figure 5.19(d) shows the contour image after employing local pairing using clouded image and the other layer. Some of the contours which are matched are shown in red circles.

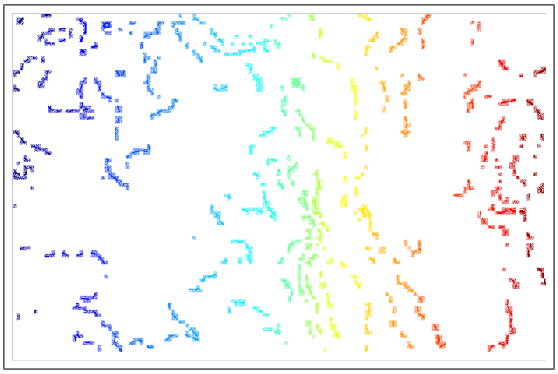
Figure 5.19 shows the results on contour layer of topomap 6 shown in Figure 3.6. It can be noted that number of gaps in these images are high owing to poor quality of topomaps (shown in Figure 3.6). The filtering process further damages the contour layer obtained by introducing more gaps.



(a) Contour Image



(b) Gap Contour Image



(c) Image after Localization

Figure 5.20: Results on Contour Layer of Topomap 6

Chapter 6

Reconstruction of Contour Layer

A contour layer is extracted from the topomap, filtered from noise and text and represented using hierarchical representation of Binary tree and graph. Digital Elevation Model(DEM), which is essential to create terrain in three dimensional space, can be interpolated using contour lines. The generation of DEM from topographic map is accomplished by multi step process. Initially the raster contour lines and its elevation are converted to vectors, and then tagged with their corresponding elevation values. The tagged vector data is then transferred to a superimposed grid to generate three dimensional surface by an interpolation or approximation algorithm. Finally it is written to standard ASCII DEM format (SYSbNI04). In this chapter various pre processing operations required to reconstruct the surface from the contour layer are elaborated. Section 6.1 deals with approximating contour segments, Section 6.2 describes Gap filling operation and finally in Section 6.3 we discuss how the represented contour layer is used to tag altitude values.

6.1 Approximation of Contour Segments

In contour line approximation or generalization, three requirements should be fulfilled (ZHJ99):

1. The contour lines must be simple, which means the number of points is reduced

2. The contour must appear smooth

3. The characteristic of contour must be retained

The contour line is represented using a binary threaded tree after segmenting into small curves, which is described in Chapter 4. Each region of the segmented contour contains portions of curve, with simple shapes and sizes compared to the original curve. Boundary-following can therefore be employed on the segment in each of these regions, and hence can be represented using various contour based approaches. Boundaries can be represented efficiently with various spline models like Bezier splines, B-splines, Hermite interpolation and rational cubic interpolation. In addition to high data reduction, this approach of using spline models has various other advantages like translation, rotation, scaling, and deformation of shapes without losing any quality (MS09).

6.1.1 Introduction

Bezier curves have been widely used for approximation in varied applications of machine vision, pattern analysis etc. Bezier curves are piecewise polynomial functions that can provide local approximation of contours using small number of parameters. Hsi-Ming Yang et al. (YLL01) has used Bezier curves for description of Chinese calligraphy characters. Asif Masood et al. (MS09) has proposed an algorithm for capturing outlines using Bezier curve approximation. Jaehwa Park et al. (PK04) have used recursive Bezier curve approximation for representation of Hand sketch graphic messages. Sarbajit Pal et al. (PGB07) have discussed how a complex digitized curve can be efficiently approximated using Cubic Bezier, and have experimentally proven that Bezier curve is more accurate and uses less number of points compared to other approximation techniques using circular arcs and/or line segments. The advantage of representing a digital curve by cubic Bezier curve is that it has relatively few control points and it enhances the quality of the semi-synthetic images by providing more natural look than approximations with circular arcs and/or line segments.

The main steps in Bezier approximation of curve include Control point identification, Bezier evaluation, stopping criterion.

6.1.2 Bezier Approximation of Contour Segments

The Bezier polynomial of degree $d-1$ has d control points. A cubic Bezier has been chosen as it provides a good trade-off between complexity and accuracy. A higher degree would increase the complexity in finding the control points and a lower degree, which is two, would not be accurate in approximating slightly complex shapes. A Cubic Bezier curve is generated with four control points, P_0 to P_3. P_0 and P_3 are the end points of curve, P_1 and P_2 are the two approximating points which control the curvature and bending of the curve. A number of techniques have been proposed in recent years, for approximating a digital curve using cubic Bezier. They aim at minimizing the approximation error and improving the computational speed. The location of control points can be accurately determined by using various search techniques like two-dimensional logarithmic search algorithm (MS09) or Genetic algorithms. A simple method of finding Bezier control points has been adopted at the cost of increasing the approximation error as our main objective is to describe how contour lines can be represented using a hierarchical representation and Bezier coordinates. However, the accuracy can be improved by segmenting the curve further as discussed.

a) Control Point Identification

A contour line is represented using a binary threaded tree (Chapter 4). The binary threaded tree stores the MBR's of the segments of contours in the leaf nodes, and the entire contour is traced using threads stored in the nodes. The starting leaf node is stored in the head node. The following procedure is adopted on each of the segments of contours contained in the MBR's of leaf nodes sequentially:

1. The curve is traced by boundary following approach.

2. The two end points of the curve are two control points, P_0 and P_3. To ensure continuity of the contour, the end point of the previous segment (i.e. previous entry in the list) should be taken as one end point, for the curve. So, distances between P_0, P_3 of both the segments are measured, and the points which are nearest are mapped to each other. As shown in Figure 6.1,

for one segment P_3 is mapped to P_3 of previous curve, whereas for the next segment P_0 is mapped to P_3.

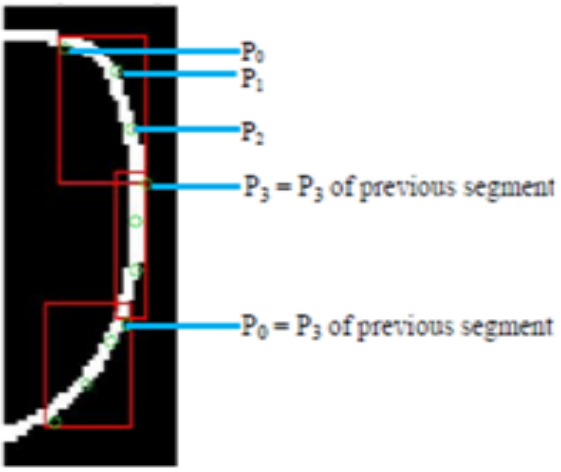


Figure 6.1: Control Points Identification

3. The other two control points are taken at one third and two third lengths of the curve respectively.

4. The length of the segment, N_p is also stored.

b) Bezier Evaluation

A Bezier curve $C(t)$ of degree n can be defined in terms of a set of control points $p_i, (i = 0, 1, 2...n)$ and is given by Bézier :

$$C(t) = \sum_{i=0}^{3} p_i B_{z_{i,n}}(t) \tag{6.1}$$

where each term in the sum is the product of a blending function $B_{z_{i,n}}(t)$ and a control point p_i. $B_{z_{i,n}}(t)$ are called Bernstein polynomials and are defined by

$$B_{z_{i,n}}(t) = {}^{n}B_i t^i (1-t)^{n-i} \tag{6.2}$$

where ${}^{n}B_i$ is the binomial coefficient.

5. A cubic Bezier is evaluated using equations 6.1 and 6.2 with the four control points obtained from steps, 2 and 3.

6. It is worth noting that the contour can be reconstructed from the four control points and the length, using equations 6.1 and 6.2. Figure 6.2 shows a contour which has been approximated using a cubic Bezier. The green curve is the curve evaluated through Bezier approximation. .

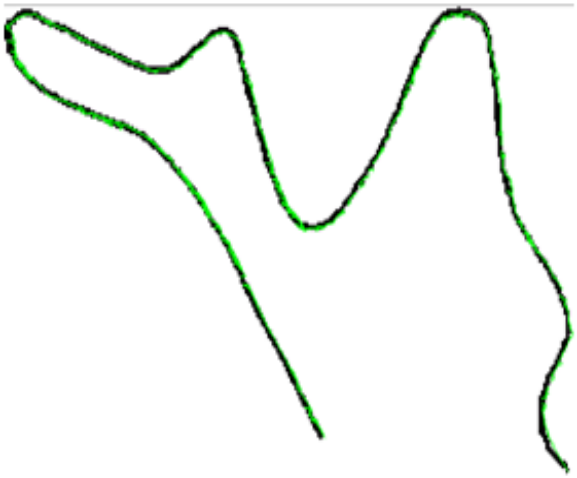


Figure 6.2: Bezier Approximation

Table 6.1 shows the Bezier control points (P_0 to P_3) of the curve.

c) Stopping Criterion

The error in representation can be minimized by computing the accuracy of fit and further updating of tree as mentioned below:

7. The accuracy of Bezier fit is estimated by computing the distances between boundary points and Bezier points. If N_p is the total number of points in a segment, and the curve points and Bezier points are denoted by C_i and B_i respectively, approximation error is calculated as the root mean square

Table 6.1: Bezier Control Points of Curve shown in Figure 6.2

P_0x	P_0y	P_1x	P_1y	P_2x	P_2y	P_3x	P_3y
151.5	237.5	137	246	150.5	239.5	165.5	239.5
151.5	237.5	130	249	120.5	249.5	115.5	248.5
91.5	232.5	98.5	240	106.5	244.5	115.5	248.5
71.5	222.5	78.5	226	85.5	230.5	91.5	232.5
49.5	218.5	55.5	220	63.5	221.5	71.5	222.5
24.5	215.5	33.5	218	41.5	219.5	49.5	218.5
24.5	180.5	4.5	194	6.5	212.5	24.5	215.5
24.5	180.5	24.5	183	26.5	180.5	28.5	180.5
38.5	174.5	32.5	178	26.5	180.5	24.5	180.5
54.5	166.5	48.5	170	43.5	172.5	38.5	174.5
68.5	156.5	62.5	162	57.5	165.5	54.5	166.5
80.5	147.5	75.5	152	69.5	155.5	68.5	156.5
78.5	126.5	84.5	135	83.5	142.5	80.5	147.5
78.5	126.5	77.5	126	80.5	128.5	84.5	131.5
42.5	110.5	53.5	115	66.5	120.5	78.5	126.5
28.5	106.5	32.5	110	39.5	111.5	42.5	110.5
14.5	88.5	10.5	98.5	17.5	104.5	28.5	106.5
25.5	63.5	24.5	72.5	20.5	80.5	14.5	88.5
12.5	32.5	16.5	43.5	21.5	55.5	25.5	63.5
8.5	1.5	8.5	26.5	7.5	18.5	12.5	32.5
12.5	32.5	43.5	44.5	46.5	55.5	54.5	65.5
54.5	65.5	55.5	69.5	60.5	75.5	65.5	78.5
65.5	78.5	68.5	82.5	73.5	86.5	79.5	89.5
79.5	89.5	79.5	90.5	86.5	95.5	93.5	100.5
93.5	100.5	99.5	104	107.5	107.5	114.5	110.5
114.5	110.5	117	113	122.5	115.5	128.5	119.5
128.5	119.5	135	124	141.5	127.5	148.5	131.5
148.5	131.5	149	132	158.5	136.5	167.5	140.5

Left child	MBR (Relative)	Data $P_0 - P_3$,N_p	Right child

Figure 6.3: Terminal / Leaf Node Structure (LN)

distance from the contour points and the Bezier curve points.

$$RMSE = \sqrt{\sum_{i=1}^{N_p} (B_i - C_i)^T (B_i - C_i)} \tag{6.3}$$

8. If the error is above a threshold, the segment can be divided further. Subsequently node of the tree, corresponding to the segment is split and the tree is updated by adding left and right child nodes to the existing node (Section 4.2).

9. A cubic Bezier is then fitted to each of the child segments. The leaf node pointers are updated.

10. This procedure can be repeated till the desired accuracy is reached.

6.1.3 Update Representation

A contour line is segmented and represented using a Binary threaded tree (Chapter 4). Each segment of the contour line is approximated using cubic Bezier, by traversing the leaf nodes of the binary threaded tree, as explained in the previous section. The control points (P_0 *to* P_3) are obtained for each segment. If the control points are stored in the tree, the contour line can be reconstructed with a certain error. Hence the data field of the leaf nodes (Section 4.3) point to the four control points and the length of the corresponding segment. Figure 6.3 shows the updated leaf node structure. Figure 6.4 shows the threaded Binary tree shown in Figure 4.3 with leaf nodes pointing to the Bezier control points obtained by approximating the segments of contour line in each leaf node.

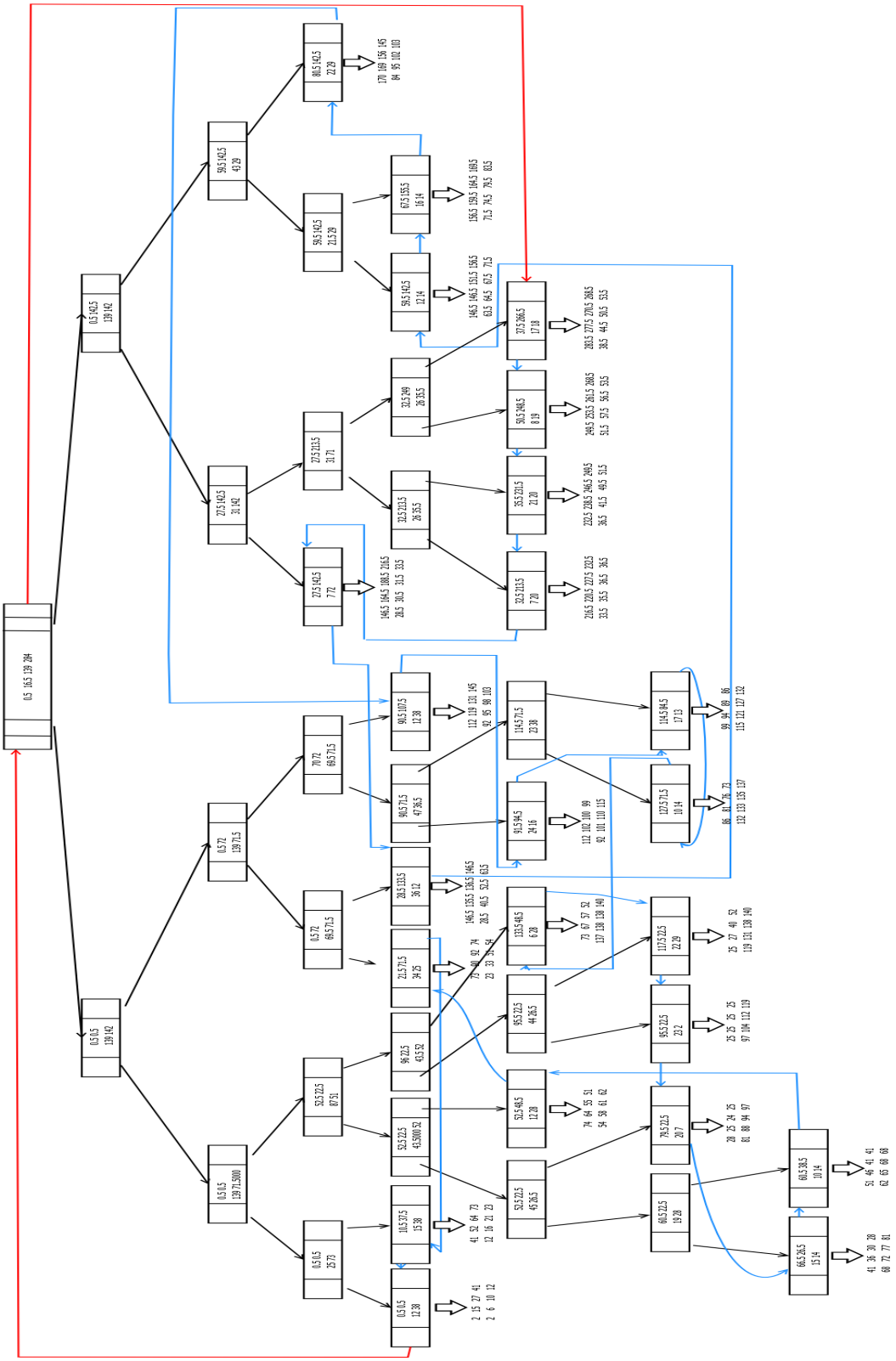


Figure 6.4: Threaded Binary Tree with Bezier Control Points

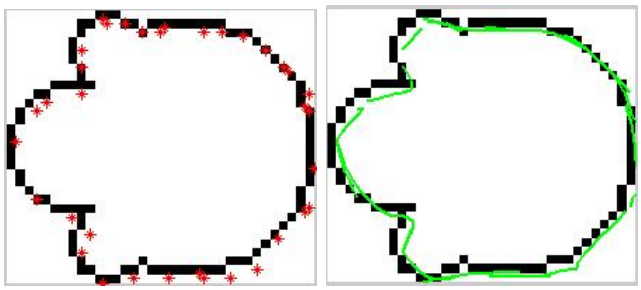
6.1.4 Results - Comparative Study

We have applied the proposed Binary tree with Bezier approximation of segments, to some of the benchmark images used for comparing various curve representation techniques like Arc tree, Strip tree etc. (Section 4.1). The chain codes of the images which have been used in this section for comparison have been provided by Dr.Biswajit Sarkar and Dr.Debranjan Sarkar. The results for the benchmark curves are from (SRS03). In the Table 6.2, n_d denotes the number of dominant points (i.e., vertices) used for approximation and d_{max} denotes the maximum deviation of any point on the curve from its approximating straight line. However we have computed d_{max} as the maximum deviation of point on the curve from its approximated point. ISE of a curve segment C which is represented by a sequence of points is computed as square of RMSE, computed in equation 6.3. Figure 6.5 shows the representation of 'Four Semicircles' curve with our method and Figure 6.6 shows the results on the Industrial parts. It can be observed from the Table 6.2 that our results are comparable to other hierarchical representation schemes. It can be seen that the number of segments required is less compared to other hierarchical representations and ISE, d_{max} can further be improved with a more accurate way of finding the Bezier control points.

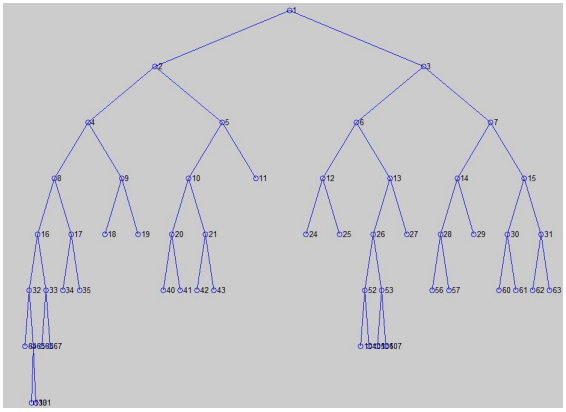
6.2 Gap Filling

6.2.1 Introduction

The contour layer extracted from topomap has gaps and discontinuities, as mentioned in the earlier chapters, due to removal of overlapping/intersecting layers. Hence reconstruction of contour layer is an essential operation in vectorizing it. Traditionally reconstruction of contour segments has been done using Image based approach, Geometric based approach or more recently Gradient based approach. Gap Identification and Matching of contour segments which are first two steps of reconstruction have been discussed in detail in Chapter 5. In this section, the last step, which is filling the gap between two matched contour segments is elaborated. Gap filling is the process of identifying the background pixels between two contour segments which should belong to the object(contour) so that the contour

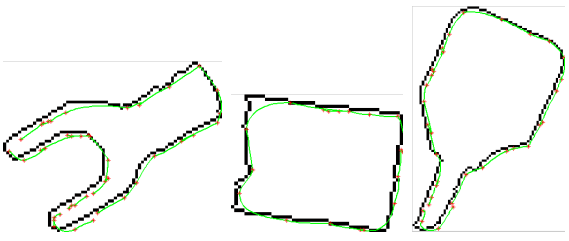


(a) Control points (b) Bezier approximation



(c) Binary tree

Figure 6.5: Representation of Semi Circle



(a) Part 1 (b) Part 2 (c) Part 3

Figure 6.6: Representation of Industrial Parts

Table 6.2: Comparison of Hierarchical Curve Representation Techniques

Part	Method	n_d	d_{max}	ISE
Semi Circle	Arc (Level 5)	33	1.414	11.814
	Strip(Level 5)	29	1.404	14.612
	Equal Error(Level 5)	30	1.000	7.289
	Proposed Method	28	0.95	10
Part 1	Arc (Level 5)	33	2.236	68.467
	Strip(Level 5)	30	2.915	91.515
	Equal Error(Level 5)	33	1.240	35.135
	Proposed Method	33	3.8	33.4499
Part 2	Arc	33	1.569	30.109
	Strip	28	1.436	28.917
	Equal Error	32	1.116	20.728
	Proposed method	16	3.7	23.3834
Part 3	Arc	33	1.789	66.934
	Strip	30	1.170	37.474
	Equal Error	33	1.612	49.921
	Proposed method	29	3.8	31.1634

line is either closed or ends at the boundary of topomap. Gap filling is attempted using two different approaches:

1. Weaving : This approach uses image properties and takes into consideration the thickness and orientation of the contour segments.

2. Bezier approximation: The contour segments have been approximated using Bezier curves and hence the gap between two such approximated contour segments is filled using the Bezier coordinates.

6.2.2 Weaving

The process of filling the gap between the identified end points is named as "Weaving", coined because of the inherent steps involved in the process, which are close to weaving of cloth. Weaving algorithm fills the gaps caused between two curves of any slope and width, in other words, it fills a region between curves. The input to the Weaving algorithm is an image with both the segments of a contour which form a gap. The image obtained after gap identification and matching is

used to isolate each gap and two segments associated with the gap. The algorithm proceeds as follows: Let the two curves be C_1 and C_2. End point of C_1 is p_1 and of C_2 is p_2, as shown in the Figure 6.7.

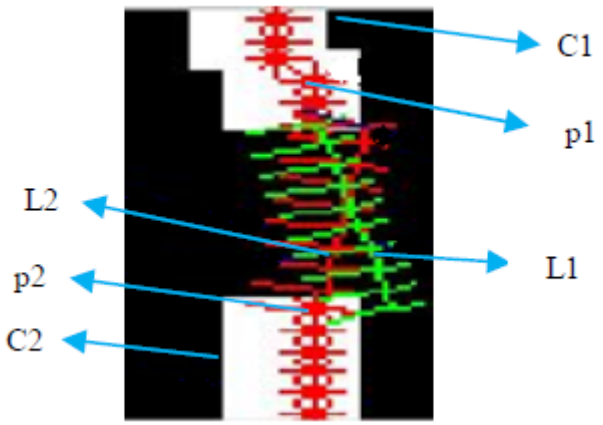


Figure 6.7: Weaving

1. Find the centre points of the first curve, say, C_1.
2. Interpolate the centre points of one curve as a line L_1 extending up to the end point of the other curve.
3. Next, slope of the line L_1 is calculated.
4. Tangents to the line L_1, whose lengths are equal to the average width of curve, are constructed at each pixel between the two end points, p_1 and p_2.
5. Steps 1) - 4) are repeated for the other curve C_2.
6. The region of intersection between both the tangents is the gap to be filled.

The steps are explained in the Figure 6.8. Figures 6.10(b), 6.10(e) show the result of applying weaving algorithm between the gaps of a portion of contour layer, which are shown in Figure 5.12.

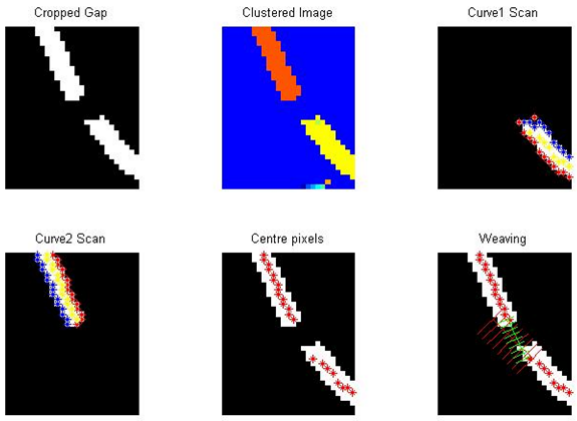


Figure 6.8: Weaving

6.2.3 Using Representation/Bezier Extrapolation

Each connected component in a contour layer is represented as a binary threaded tree and a cubic Bezier is fit to the segments of the associated curve. From each tree, a list of leaf nodes which can trace the curve, and four Bezier control points are computed.

Identification of end segments of contour lines using the tree has been explained in Chapter 4. Once the end segments of all the curves have been identified, matching techniques can be employed to match the two segments that are required to be connected. The gaps between such end segments are filled using Bezier extrapolation.

1. The end segments identified are approximated using Bezier curves and hence have four control points each, out of which two are end points of the curves.

2. The end points of the curves are the two control points P_0 and P_3 for the Bezier curve in the gap.

3. The other two control points P_1 and P_2 are computed by finding the tangents at the control points P_0 and P_3, and taking a point at a length $l/3$ on the tangents, where l is the distance between two control points P_0 and P_3.

4. Hence we have four control points P_0 to P_3 and the approximate length of the gap, l using which a Bezier curve that fills the gap is computed.

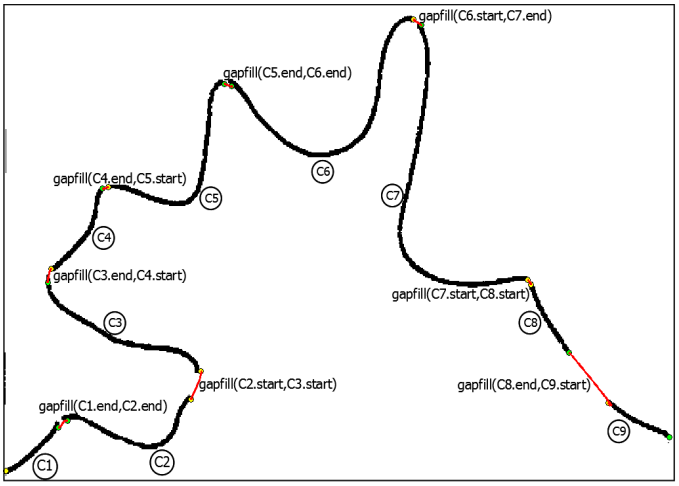


Figure 6.9: Gap filling of Contour

Figure 6.9 shows the gap filling between segments of a contour. The Figures 6.10(c), 6.10(f) show the result of applying Bezier interpolation between the gaps of a portion of contour layer. It can be observed that Bezier interpolation only fills the gap with single pixel width line whereas Weaving fills the area of gap with a thick line. However gap filling using Bezier interpolation can be an on-line process.

6.3 Tagging Altitude Values

The contour layer has been reconstructed, represented using a binary threaded tree and each segment approximated using Bezier curves. The altitude values filtered from the contour lines have been recognized using LeNet - 5. To generate a 3D model from the contour lines, the altitude values have to be tagged to the contour lines. This is accomplished using the binary tree representation of lines. Finding the contour lines which pass through a altitude tag, is similar to region query explained in Section 4.5 of Chapter 4. The MBR of altitude tag is found, and the contour lines which intersect with the MBR should be found. When more than one contour is obtained, the contour whose tree intersects with the MBR of altitude tag at a deeper level is chosen. Figure 6.11 shows the results on two images of contour layers and the corresponding altitude tag layers.

(a) Sample Image (b) Weaving result (c) Bezier interpolation result

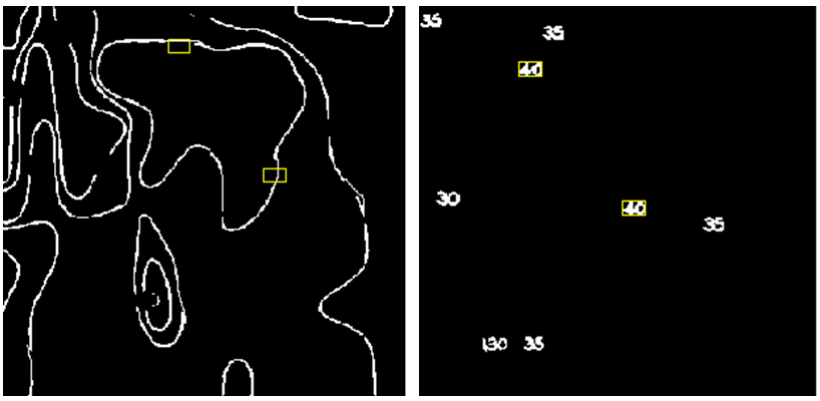
(d) Sample Image (e) Weaving result (f) Bezier interpolation result

Figure 6.10: Comparison of Gap Filling

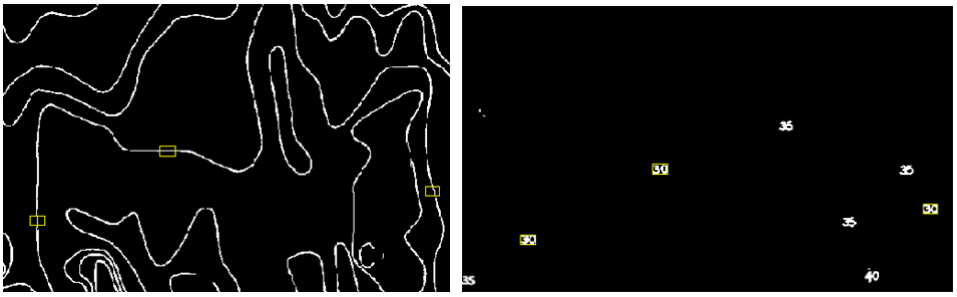
6.4 Results

Figure 6.12 shows the results of applying all the steps of Reconstruction i.e Gap Identification, Matching and Gap Filling on a contour layer extracted from topomap 1(Figure 3.6).

Figure 6.13 shows the result of applying Gradient based reconstruction proposed by (JS07) on the contour layer shown in Figure 6.13(c). (The image is enlarged to show details as it is a thinned image). Gaps filled are shown in green. The algorithm is applied by using AutoDEM software (Aut), which is a freeware GIS software. As can be observed from the figure, the algorithm operates on a thinned image, whereas for our Gap Identification and Weaving algorithms, it is not necessary to thin the image. Also the algorithm fails in some cases, shown in red circles where Weaving has worked. The cases where Weaving has failed and Gradient algorithm has worked are shown in yellow circles. Weaving algorithm fails to fill gaps in the cases where it is not possible to interpolate one contour segment to the other segments end point.

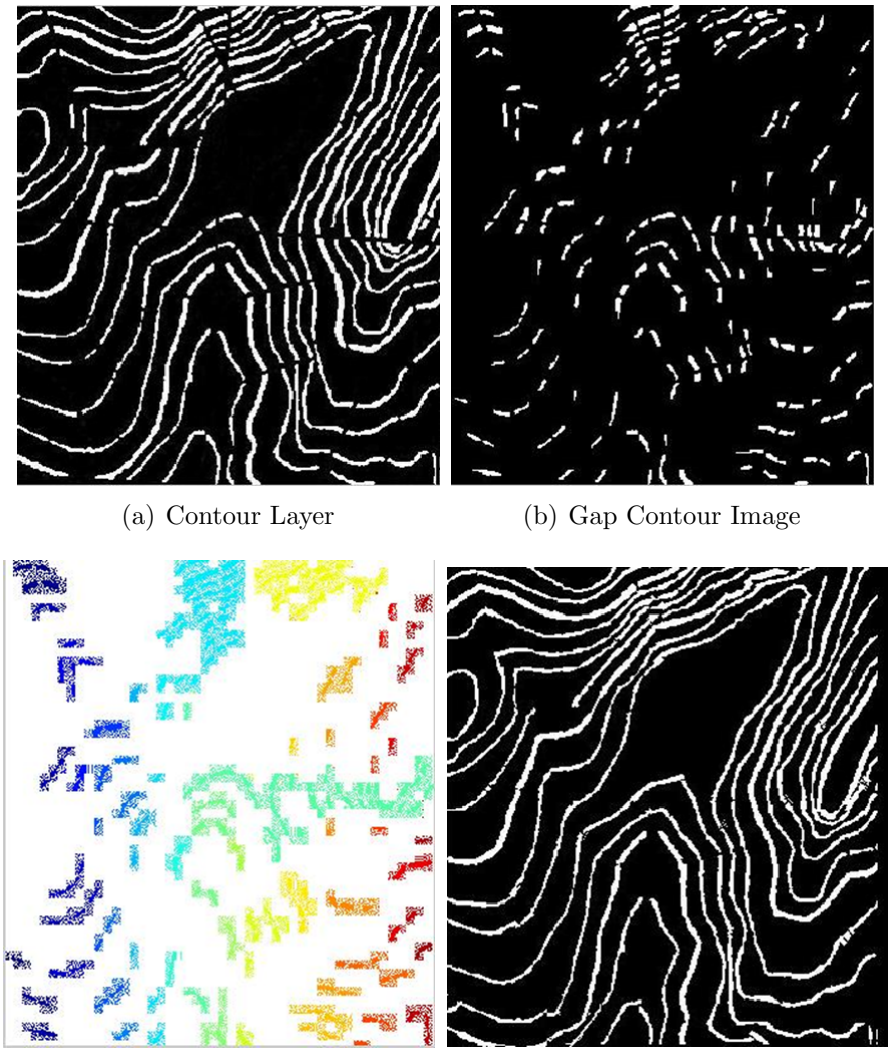


(a) Contour Layer (b) Altitude Tag Layer



(c) Contour Layer (d) Altitude Tag Layer

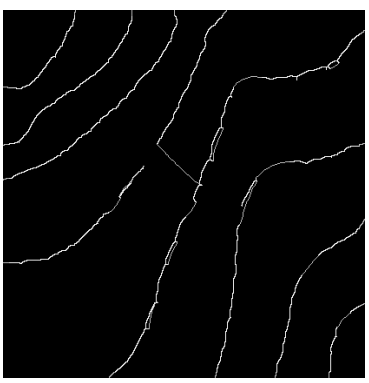
Figure 6.11: Altitude Value Tagging



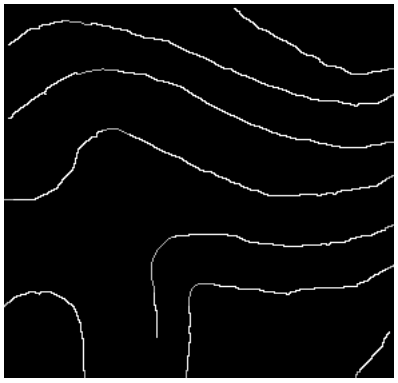
(a) Contour Layer (b) Gap Contour Image

(c) Image after Localization (d) Gap Filling using Weaving

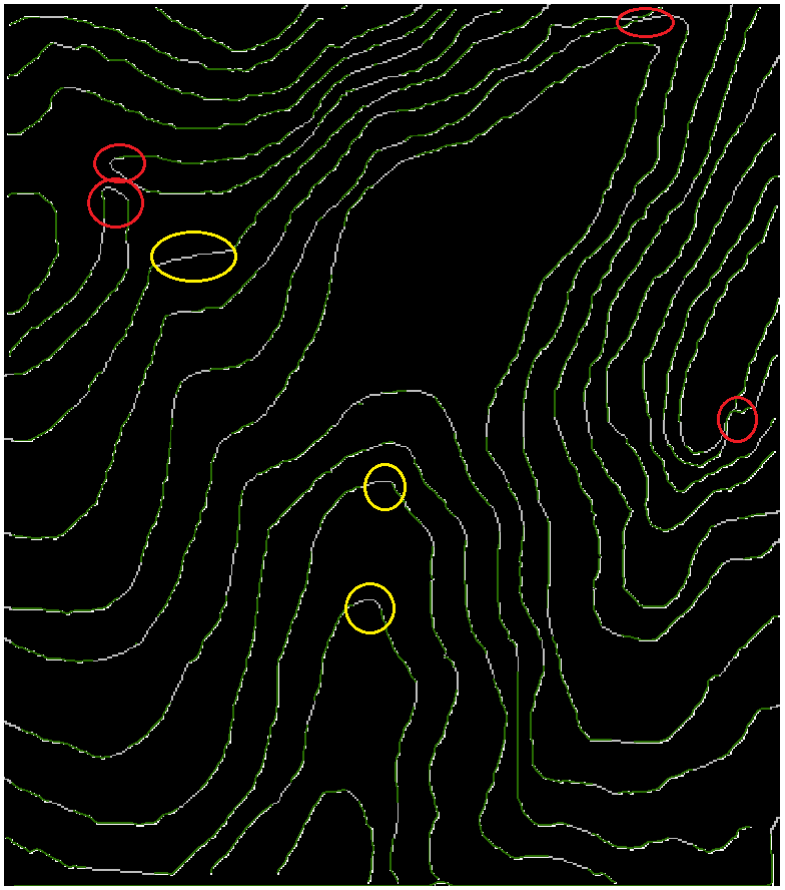
Figure 6.12: Reconstruction Of Contour Layer



(a) Portion of Contour Layer (6.10(a))



(b) Portin of Contour Layer (6.10(d))



(c) Contour Layer

Figure 6.13: Comparison of Gap Filling

Chapter 7

Conclusions and Future Directions

7.1 Conclusions

Based on extensive literature survey we have been able to conclude that research on fully automatic feature extraction and object recognition from scanned topographic maps deals with two main difficulties: automatic extraction of layers and accurate recognition. There are two critical issues for segmentation of topographic maps: what segmentation method should be utilized and what colour space should be adopted? Our survey indicates that the majority of colour segmentation approaches are based on monochrome segmentation approaches operating in different colour spaces. All the existing colour segmentation algorithms are strongly application dependent, in other words, there are no general algorithms and colour space that are good for all colour images. A summary of all existing pre-processing, segmentation and post-processing methods has been furnished in tabular form. Current contour layer vectorization approaches, based upon image, geometry and gradient vector flow, have been described. In this work, we have proposed a systemic approach, comprising various major and minor steps of the information flow, which encompasses the wide gamut of algorithms employed in topomap processing. The methodology we have proposed, divides the task into three major stages, namely Extraction, Representation and Reconstruction, along with the associated sub-tasks in each stage. With the overall design in

place, we have set out on identifying suitable algorithms to accomplish each task or sub-task. Wherever convenient we have adapted existing algorithms, which have proven record of performance in other image processing applications, to perform the required task. In other cases, we have devised our own algorithms to achieve improved performance.

With respect to extraction of contour layer, our focus has been more on identifying the most suitable colour feature set which can segment any kind of topomap, rather than trying to improvise the segmentation algorithm further. We have meticulously evaluated the advantages and disadvantages of well known colour spaces. We alleviate the tricky problem of choosing the right colour space by addressing the issue as a feature selection problem, thereby selecting best features from all the colour spaces for any given image. Feature selection is done using Principal Component Analysis (PCA). Quantization followed by clustering, based on peak climbing approach, and merging of clusters completes the extraction of contour layer. The extracted contour layer is subjected to filtering to remove noise and altitude tags attached to the contour lines.

The possibility of using traditional boundary or region based approaches for contour line representation has been explored. Due to the existence of noisy contour lines in topomaps, region based methods of representation have a clear edge over boundary based methods like thinning etc. We have proposed a tree based representation obtained from segmenting the contour lines in the extracted binary contour layer. The tree, which is designed in such a way that it provides a hierarchical coarse to fine representation of the contour, captures the contour segment characteristics in terms of the associated number of MBRs. From the traversal of each tree, a list of leaf nodes which can trace the contour is computed. If the topological characteristics of contour lines are preserved, the entire contour layer is represented by binary threaded trees of each contour line. A relational model which stores the relation between various contour line segments has been proposed. Such a model is required to facilitate filling of gaps during reconstruction and also for linking different parts of a topomap distributed across various nodes in a parallel architecture. Application of queries to the relational model has also been demonstrated.

The process of extraction of contour layer followed by filtering of overlapping layers is a lossy one and results in gaps in the extracted layer. Reconstruction is the process of identifying and filling of gaps in a binary contour layer. The limitations of traditional line tracking methods and methods specific to contour layers like image or geometric based, which could be either local or global, with or without the use of external information, have been brought out. A new approach has been suggested which is based on detection of gap endpoints, thereby making the reconstruction process fully automated. The gap endpoints are subsequently paired using the well-known expectation maximization algorithm which has led to significant reduction in the computational complexity of the matching process. Pairing of endpoints is followed by filling of the gap, which is independent of the thickness and orientation of contour lines, by virtue of a new approach suggested. The recursive version of the algorithm has also been proposed which can be implemented on a parallel architecture.

Publications

Following is the list of publications:

1. Abstract on " Topographic map processing on GEON ", Proceedings of AP Science Congress, 2008

2. " Automatic gap identification towards efficient contour line reconstruction in topographic maps." Asia International Conference on Modelling and Simulation, p:309 - 314, 2009

3. " Adaptive Altitude Feature Recognition for Paper based Topographic map ", The 4th Mahasarakham International Workshop on Artificial Intelligence, Thailand, p:2 - 11, 2010

4. " Contour layer Extraction from Colour Topographic Map by Feature Selection approach", IEEE Symposium on Computers and Informatics, Kuala Lumpur, Malaysia, 2011, p: 425 - 430, 2011

5. " Representation of Contour lines Extracted from Topographic map ", communicated to Computers & Geosciences - Elsevier

7.2 Future Directions

The algorithms proposed for the topomap processing can be fine tuned and efficiency in terms of execution time and memory can be addressed. In the process of extraction, various feature selection techniques can be experimented, to ensure that all the layers are extracted accurately. Recognition of altitude tags requires much more work as the reported techniques need to be further studied with larger training sets. Our research not being specifically focused on spatial databases and query processing, the contour layer representation scheme that has been proposed, although efficient, needs to be extended to address a query processing system. One possible improvement could be using binary threaded trees to represent the linear as well as area features. Thus, only the data to be stored in different layers varies. As a future task, the width of the contour lines can be extracted from the representation scheme proposed in this work. This information can be obtained from the ratio of approximated contour segment to the MBR area, utilizing the knowledge of the threshold to split a node based on density ratio. With the help of Bezier approximation the width of the curve can subsequently be modelled. The Reconstruction algorithm that we have put forward is computationally expensive, therefore, various parallel computing paradigms need to be explored and applied. In addition, the performance of the proposed reconstruction algorithms has been evaluated mostly based on visual quality of the output images. This is due to the fact that there exists neither a standard set of images on which the algorithms could be tested, nor approved quantitative measures of evaluation to compare the proposed algorithm against existing algorithms. Future efforts in this direction could be to perform a quantitative evaluation of the algorithms for contour line extraction and vectorization. Other long-term goals could be - development of algorithms for extraction and digitization of different topomap layers, design of a knowledge based system for topomap processing, building of a 3D model from digitized contour layer and finally, grid based implementation of the algorithms. Extension of the algorithms proposed in this work to extraction of layers in any general colour document could open up new research avenues.

References

[AAC09] Jeffrey Armstrong, Maher Ahmed, and Siu-Cheung Chau. A rotation-invariant approach to 2d shape representation using the hilbert curve. *ICIAR 2009, LNCS*, 5627:594–603, 2009. 80

[ABE98] N. Amenta, M. Bern, and D. Eppstein. The crust and the beta-skeleton: Combinatorial curve reconstruction. *Graphical models and image processing*, 60(2):125 – 135, 1998. 21, 22

[AK05] C.P.Lo Albert and K.W.Yeung. *Concepts and techniques of Geographic Information Systems.* PH Series in Geographic Information Science, Prentice-Hall, India, 2005. 1, 2, 27

[ALR77] A.P.Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Roy. Statistics Society*, 39(1):1–38, 1977. 125

[AM09] Soycan A and Soycan M. Digital elevation model production from scanned topographic contour maps via thin plate spline interpolation. *The Arabian Journal for Science and Engineering Science*, 34(1A):121 – 134, 2009. 22

[AS99] Patrice Arrighi and Pierre Soille. From scanned topographic maps to digital elevation models. *Proceedings of Geovision, International Symposium on Imaging Applications in Geology*, 1999. 20, 21, 24, 103

[ASL90] M. Ansoult, P. Soille, and J. Loodts. Mathematical morphology: A tool-for automated gis data acquisition from scanned thematic maps. *Photogramm.Eng. Remote Sens*, 56(9):1263 –1271, 1990. 29

[Aut] AutoDEM. http://www.autodem.com/. 147

[Bal81] Dana H. Ballard. Strip. trees:a hierarchical representation for curves. *Graphics and Image Processing, Communications of the ACM*, 24(5), 1981. 80

[BHKS93] Thomas Brinkhoff, Holger Horn, Hans-Peter Kriegel, and Ralf Schneider. A storage and access architecture for efficient query processing in spatial database systems. *Advances in Spatial Databases, Lecture Notes in Computer Science*, 692:357–376, 1993. 95

[BK92] B.K.Ray and K.S.Ray. An algorithm for detection of dominant points and polygonal approximation of digitized curves. *Pattern Recognition Letters*, 13:849–856, 1992. 80

[BKKR07] Nadya T. Bliss, Jeremy Kepner, Hahn Kim, and Albert Reuther. Pmatlab: Parallel matlab library for signal processing applications. *ICASSP, IEEE*, pages 1189 – 1192, 2007. xiii, 119, 121

[Cen98] Jorge Silva Centeno. Segmentation of thematic maps using colour and spatial attributes. *Selected Papers from the Second International Workshop on Graphics Recognition, Algorithms and Systems*, pages 221–230, 1998. 15, 18

[CEO05] Ron Choy, Alan Edelman, and Cleve Moler Of. Parallel matlab: Doing it right. *Proceedings of the IEEE*, pages 331 – 341, Feb 2005. 117

[CJSW01] H.D. Cheng, X.H. Jiang, Y. Sun, and Jingli Wang. Color image segmentation: advances and prospects. *Pattern Recognition*, 34:2259 – 2281, 2001. 14, 30

[CR88] C.H.Teh and R.T.Chin. On image analysis by the methods of moments. *IEEE Trans. Pattern Anal. Mach. Intell PAMI*, 10(4):496–513, 1988. 80

[CWQ06a] Yang Chen, Runsheng Wang, and Jing Qian. Extracting contour lines from common-conditioned topographic maps. *IEEE TRANSACTIONS ON GEO SCIENCE AND REMOTE SENSING*, 44(4):1048–1057, 2006. 15, 17, 21, 102

[CWQ06b] Yang Chen, Runsheng Wang, and Jing Qian. Extracting contour lines from common-conditioned topographic maps. *IEEE Transactions on Geoscience and Remote Sensing*, 44(4):1048 – 1057, April 2006. 21, 24, 29

[D89] Rhind D. 'Why GIS?'. *ARC News*, 11(3), 1989. CA: Environmental Systems Research Institute,Inc. 1

[DP05] D.Gil and P.Radeva. Extending anisotropic operators to recover smooth shapes. *Computer Vision and Image Understanding*, 99(1):110–125, 2005. 22

[DRM99] T. K. Dey, E. A. Ramos, and K. Mehlhorn. Curve reconstruction:connecting dots with good reason. *In Proceedings of the 15th Symposium on Computational Geometry*, page 197 206, 1999. 22

[D.S93] D.Sarkar. A simple algorithm for detection of significant vertices for polygonal approximation of chain-coded curves. *Pattern Recognition Letters*, 14:959–964, 1993. 80

[DW00] T. K. Dey and R. Wenger. Reconstruction curves with sharp corners. *SCG 00: Proceedings of the sixteenth annual symposium on Computational geometry*, page 233 241, 2000. 22

[ELA94] N. Ebi, B. Lauterbach, and W. Anheier. An image analysis system for automatic data acquisition from colored scanned maps. *Machine Vision Applications*, 7(3):148 – 164, 1994. 19, 29

[FD82] Leberl F. and Oslon D. Raster scanning for operational digitizing of graphical data. *Photogrammetric Engineering and Remote Sensing*, 48(4):615 – 627, April 1982. 20

[FK88] L.A. Fletcher and R. Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Transactions on pattern Analysis and Machine Intelligence*, 10:910 – 918, 1988. 53, 56

[Gan06] Pankaj Ganguly. Modified arc tree based hierarchical representation of digital curve. *Pattern Recognition Letters*, 27:529–535, 2006. 80

[GB11] Tudor GHIRCOIAS and Remus BRAD. Contour line extraction and reconstruction from topographic maps. *Special issue of The Romanian Educational Network - RoEduNet*, Jan 2011. 22, 24

[GK79] G.M.Hunter and K.Steiglitz. Operations on images using quadtrees. *IEEE Transactions on PAMI*, 1(2):145–153, 1979. 80

[Gol99] Christopher Gold. Crust and anti-crust: a one-step boundary and skeleton extraction algorithm. *Proceedings of the fifteenth annual Symposium on Computational Geometry, SCG*, pages 189 – 196, 1999. 22

[GR88] Alan Gibbons and Wojciech Rytter. *Efficient Parallel Algorithms*. Cambridge University Press, 1988. 122

[Gre87] D. D. Greenlee. Raster and vector processing for scanned linework. *Photogrammetric Engineering and Remote Sensing*, 53:1383–1387, Oct 1987. 20

[GW90] Oliver Gunther and Eugene Wong. The arc tree :an approximation scheme to represent arbitrary curved shapes. *Comput.Vision Graphics Image Process.*, 51:313–337, 1990. 80

[HCY08] Xiaoan Tang Hong Chen and Yaoming Yang. Recognition of contour line from scanned military topographic maps. *IEEE International Conference on Information and Automation*, pages 873 – 878, June 2008. 15, 21, 24

[HGI95] Luo H.Z, Agam G, and Dinstein I. Directional mathematical morphology approach for line thinning and extraction of character strings from maps and line drawings. *International Conference on Document Analysis and Recognition(ICDAR)*, 1:257, 1995. 54

[HL09] Thomas C. Henderson and Trevor Linton. Raster map image analysis. *10th International Conference on Document Analysis and Recognition, IEEE*, pages 376–380, 2009. 9, 15, 16

[HS98] Ellis Horowitz and Sartaj Sahni. *Fundamentals of Computer Algorithms.* Galgotia Publications, India, 1998. 85

[I.G82] I.Gargantini. An effective way to represent quadtrees. *Comm. ACM*, 25(12):905–910, 1982. 80

[IJ84] I.M.Anderson and J.C.Bezdek. Curvature and tangential deflection of discrete arcs: a theory based on the commutator of scatter matrix pairs and its application to vertex detection in planar shape data. *IEEE Trans Pattern Anal.Machine Intell*, 6:27–40, 1984. 80

[JAR98] J.M.Squyres, A.Lumsdaine, and R.L.Stevenson. A toolkit for parallel image processing. *Proceedings of the SPIE Conference on Parallel and Distributed Methods for Image processing*, pages 69 – 80, 1998. 117

[JE94] J.C.Perez and E.Vidal. Optimal polygonal approximation of digitized curves. *Pattern Recognition Letters*, 15:743–750, 1994. 80

[JS04] J.Kepner and S.Ahalt. Matlabmpi. *Journal of Parallel and Distributed Computing*, 64(8):997 – 1005, 2004. 119

[JS07] J.Pouderoux and S.Spinello. Global contour lines reconstruction in topographic maps. *Ninth International Conference on Document Analysis and Recognition*, 2:779–783, 2007. 20, 22, 55, 77, 147

[JY04] Du Jinyang and Zhang Yumei. Automatic extraction of contour lines from scanned topographic map. *IEEE*, pages 2886–2888, 2004. 24, 103, 105

[KB90] Alireza Khotanzad and Abdelmajid Bouarfa. Image segmentation by a parallel, nonparametric histogram based clustering algorithm. *Pattern Recognition*, 23(9):961 – 973, 1990. 29, 36, 39

[KGG02] Rangachar Kasturi, Lawrence O Gorman, and Venu Govindaraju. Document image analysis: A primer. *Sadhana*, 27(1):3 – 22, February 2002. xi, 8

[KSL+02] Tombre K, Tabbone S, Peissier L, Lamiroy B, and Dosch P. Text /graphics separation revisited. *In Document Analysis Systems V*, 2423:615 – 620, 2002. 54

[KZ03] Alireza Khotanzad and Edmund Zink. Contour line and geographic feature extraction from usgs color topographical paper maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1), 2003. 15, 18, 21, 24, 28, 29, 81

[LB10] Stefan Leyk and Ruedi Boesch. Colors of the past: color image segmentation in historical topographic maps based on homogeneity. *Geoinformatica*, 14:1–21, 2010. 14, 15, 16, 45

[LKH95] Eikvil L, Aas K, and Koren H. Tools for interactive map conversion and vectorization. *Third International Conference on Document Analysis and Recognition,ICDAR*, pages 927 – 930, 1995. 20

[LMR96] Rick L. Lawrence, Joseph E. Means, and William J. Ripple. An automated method for digitizing color thematic maps. *Journal of the American Society of Photogrammetric Engineering and Remote Sensing*, 62(11):1245 – 1248, November 1996. 1

[LNSF03] Cheng-Lin Liu, Kazuki Nakashima, Hiroshi Sako, and Hiromichi Fujisawa. Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recognition*, 36:2271 – 2285, 2003. 68

[LS93] D.S. Lee and S.N. Srihari. Hand printed digit recognition: a comparison of algorithms. *Proceedings of the Third International Workshop on Frontiers of Handwriting Recognition*, pages 153 – 164, 1993. 68

[Map11] Northwest Maps. http://www.nwmaps.com/. April 2011. 3

[MMEA88] Musavi M, Shirvaikar M, Ramanathan E, and Nekovei A. A vision based method to automate map processing. *Pattern Recognition*, 21(4):319 – 326, 1988. 20

[MS92] Haralick Robert M., , and Linda G. Shapiro. *Computer and Robot Vision.* Volume I, Addison-Wesley, 1992. 56

[M.S02] M.Salotti. Optimal polygonal approximation of digitized curves using the sum of square deviation criterion. *Pattern Recognition*, 35:435–443, 2002. 80

[MS09] Asif Masood and Muhammad Sarfraz. Capturing outlines of 2d objects with bezier cubic approximation. *Image and Vision Computing*, 27(6):704–712, 2009. 134, 135

[P.C97] P.Cornic. Another look at the dominant point detection of digitized curves. *Pattern Recognition Letters*, 18(4):13–25, 1997. 80

[PGB07] Sarbajit Pal, Pankaj Ganguly, and P.K. Biswas. Cubic bezier approximation of a digitized curve. *Pattern Recognition*, 40:2730 – 2741, 2007. 134

[PGPG07] Joachim Pouderoux, Jean Christophe Gonzato, Aurelien Pereira, and Pascal Guitton. Toponym recognition in scanned color topographic maps. *International Conference on Document Analysis and Recognition(ICDAR)*, pages 531 – 535, September 2007. 53

[PK04] Jaehwa Park and Young-Bin Kwon. An efficient representation of hand sketch graphic messages using recursive bezier curve approximation. *Image Analysis and Recognition,Lecture Notes in Computer Science*, 3211:392–399, 2004. 134

[PKA+09] Ratika Pradhan, Shikhar Kumar, Ruchika Agarwal, Mohan P. Pradhan, and M. K. Ghose. Contour line tracing algorithm for digital topographic maps. *International Journal of Image Processing (IJIP)*, 4(2):156–163, 2009. 22, 24

[PT08] Aria Pezeshk and Richard L. Tutwiler. Contour line recognition and extraction from scanned colour maps using dual quantization of the intensity image. *IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, pages 173 – 176, March 2008. 9, 14, 16, 36

[PVL+08] P.P.Roy, Eduard Vazquez, Josep Llados, Ramon Baldrich, and Umapada Pal. A system to segment text and symbols from color maps. *7th International Workshop on Graphics Recognition Recent Advances and New Opportunities: LNCS*, 5046:245 – 256, 2008. 54

[P.Y99] P.Y.Yin. Genetic algorithms for polygonal approximation of digital curves. *Int. J. Pattern Recognition Artificial Intell*, 13:1061–1082, 1999. 80

[P.Y03] P.Y.Yin. Ant colony search algorithms for optimal polygonal approximation of plane curves. *Pattern Recognition*, 36:1783–1797, 2003. 80

[RA05] Carol Rus and Jaakko Astola. Legend based elevation layers extraction from a color-coded relief scanned map. *IEEE International Conference on Image Processing (ICIP)*, pages 237–240, 2005. 8, 21

[RBO07] Romain Raveaux, Jean-Christophe Burie, and Jean-Marc Ogier. A colour document interpretation: Application to ancient cadastral maps. *International Conference on Document Analysis and Recognition(ICDAR)*, pages 1128 – 1132, September 2007. 15, 17

[RC01] Cao R and Tan C.L. Text/graphics separation in maps. *Proceedings of 4th IAPR International Workshop on Graphics Recognition*, 1:44 – 48, September 2001. 54

[RIC10] Samet R, Askerzade Askerbeyli I.N, and Varol C. An implementation of automatic contour line extraction from scanned digital topographic maps. *Appl. Comput. Math.*, 9(1):116 – 127, 2010. 21, 24

[RM00] Vitorino Ramos and Fernando Muge. Map segmentation by colour cube genetic k-mean clustering. *The Proceeding of 4th European Conference on Research and Advanced Technology for Digital Libraries,LNCS*, 1923:319323, 2000. 15, 18

[RRA04] Carol Rus, Corneliu Rusu, and Jaakko Astola. Elevation contours extraction from a colour coded relief scanned map. *IEEE International Conference on Image Processing (ICIP)*, pages 1699 – 1702, 2004. 15, 17, 24, 29

[Sam84] H. Samet. The quadtree and related hierarchical data structures. *ACM Comput. Surveys*, 16(2):187–260, 1984. 80

[SG04] Spinello Salvatore and Pierre Pascal Guitton. Contour line recognition from scanned topographic maps. *Proceedings of Winter School of Computer Graphics: WSCG,*, pages 419 – 426, February 2004. 20, 22, 24, 105

[SRS03] Biswajit Sarkar, Sanghamitra Roy, and Debranjan Sarkar. Hierarchical representation of digitized curves through dominant point detection. *Pattern Recognition Letters*, 24:2869–2882, 2003. 80, 141

[Sue92] C.Y. Suen. Computer recognition of unconstrained handwritten numerals. *Proc.IEEE*, 80(7):1162 – 1180, 1992. 68

[SY99] S.C.Huang and Y.N.Sun. Polygonal approximation using genetic algorithms. *Pattern Recognition*, 32:1409–1420, 1999. 80

[SYSbNI04] Loh Mun San, Safie Mat Yatim, Noor Azam Md Sheriff, and Nik Isrozaidi bin Nik Ismail. Extracting contour lines from scanned topographic maps. *Proceedings of the International Conference on Computer Graphics, Imaging and Visualization*, 2004. 24, 106, 133

[TC84] T.Y.Zhang and C.Y.Suen. A fast parallel algorithm for thinning digital patterns. *Comm ACM*, 27:236 – 239, 1984. 106

[TC98] Karl Tombre and Atul K. Chhabra. *Graphics Recognition: Algorithms and Systems*, volume 1389. Lecture Notes in Computer Science, Springer-Verlag, 1998. 13

[TR87] Amin T and Kasturi R. Map data processing: Recognition of lines and symbols. *Optical Engineering*, 26(4):354 – 358, 1987. 20

[Wat00] Toyohide Watanabe. Recognition in maps and geographic documents: Features and approach. *Selected Papers from the Third International Workshop on Graphics Recognition, Recent Advances,GREC*, 1941:39–49, 2000. xi, 6, 7

[Wis95] Stephen Wise. Scanning thematic maps for input to geographic information systems. *Computers Geosciences*, 21(1):7 – 29, 1995. 21

[WY00] W.Y.Kim and Y.S.Kim. A region-based shape descriptor using zernike moments. *Signal Processing: Image Communication*, 16(1-2):95–102, 2000. 80

[XH04] Zhou X.Z and Zhen H.L. Automatic vectorization of contour lines based on deformable model and field flow orientation. *Chiense Journal of Computers*, 8:1056 – 1063, 2004. 22

[XZZ06] Dongjun Xin, Xianzhong Zhou, and Huali Zheng. Contour line extraction from paper-based topographic maps. *Journal of Information and Computing Science*, 1(5):275–283, 2006. 22, 24, 105

[Y.L95] Y.LeCun. Comparison of learning algorithms for handwritten digit recognition. *F. Fogelman-Soulie,P.Gallinari (Eds.), Proceedings of the International Conference on Artificial Neural Networks*, pages 53 – 60, 1995. 68

[YLL01] Hsi-Ming Yang, Jainn-Jyh Lu, and Hsi-Jian Lee. A bezier curve-based approach to shape description for chinese calligraphy characters. *Sixth International Conference on Document Analysis and Recognition*, pages 276 – 280, 2001. 134

[YLYP98] LeCun Y., Bottou L., Bengio Y., and Haffner P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278 – 2344, November 1998. 69, 72

[YMSA08] Y.Ebrahim, M.Ahmed, S.C.Chau, and W. Abdelsalem. Shape representation and description using the hilbert curve. *Pattern Recognition Letters*, 30(4):345–358, 2008. 80

[ZHJ99] Li Z, Sui H, and Gong J. A system for automated generalization of contour lines. *In Proceedings of International Cartographic Conference Ottawa,Canada*, 1999. 133

Synopsis of the thesis

A Systemic Approach to Topographic Map Processing

Submitted by

Sandhya Banda

(Reg. No. 06MCPC05)

Under the supervision of

Prof. Arun Agarwal and Dr. Rajeev Wankar

for the award of the degree of

Doctor of Philosophy

in

Computer Science

Department of Computer and Information Sciences

School of Mathematics and Computer and Information Sciences

University of Hyderabad

Hyderabad - 500046

INDIA

NOVEMBER, 2010

1 Introduction

Automatic spatial feature extraction and recognition of cartographic documents for Geographical Information System (GIS) represents a research challenge at the intersection of GIS and Image processing. An informal definition of GIS has been given as ([D89]): "...a system of hardware, software and procedures designed to support the capture, management, manipulation, analysis, modelling, and display of spatially referenced data for solving complex planning and management problems". A fundamental characteristic of a GIS is its ability to handle spatial data, i.e. locations of objects in geographical space, and the associated attributes. The first step in the building of a GIS is data acquisition. A map which is a graphical representation of the spatial structure of the physical and cultural environments is one of the primary sources of data in a GIS. This work relates to a class of topographic maps that describe the terrain features.

1.1 Topographic Map

A topographic map gives a detailed and accurate graphic representation of cultural and natural features of the terrain based on topological surveys [AK05]. Topographic maps commonly known as topomaps are essential tools for spatial analysis with respect to various activities like infrastructure planning, construction, mining, disaster management and recovery, town planning, organization of roads and exploitation of natural resources. The information on these maps consists of sets of linear features and area features. Linear features are lines on the topographic map that convey geographic information about the locations of rivers, roads, and political boundaries, whereas, area features are regions on the topographic map that portray information about the location and placement of buildings, vegetation, and bodies of water. Topomaps describe spatial data using entities, attributes and relationships in an "unstructured representation. Entities on maps are described by point, line, and area objects. Symbols, Text, Colour are commonly

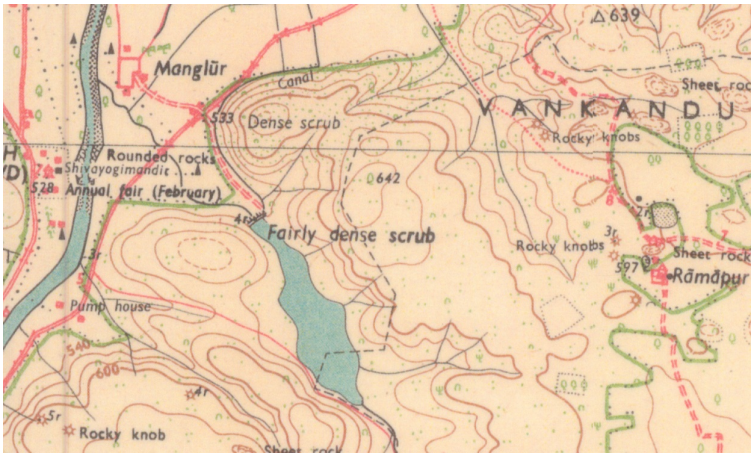


Figure 1: Sample Topomap

used on maps to represent attributes associated with the entities. Relationships among entities are depicted visually. Topographic maps are available from many suppliers, each supplier specifying standards for linear and area feature attribute. These standards specify everything from the color and thickness of a linear feature to the shape and color of an area feature. One of the largest suppliers of topographic maps is the United States Geological Survey (USGS), available for the entire territory of the United States in several map scales from 1:24,000 to 1:500,000. In India, topomaps of the regions confirm to the standards of Survey Of India. A sample map (part of 1:50000, 48M/13 First Edition) prepared according to standards of Survey Of India, is shown in Figure 1. In this map, water bodies are shown in blue, roads in red, toponyms in black, contour lines in brown. Topographic maps use very large map scales and very thin feature geometry to convey as much terrain detail as possible. At a typical sampling resolution of 300 dots per inch, a 3' X 4' map would yield an image of size 10800 X 14400 pixels. Because of the large number of measurements involved, spatial analysis based on maps has been a very time-consuming and tedious task.

1.2 Contour layer

The most distinguishing feature on the topographic map are contour lines. Contour lines are curve features that connect contiguous points of the same terrain elevation (altitude). They convey three dimensional information of the terrain's surface. It is important to note that without these contour lines, a topographic map degenerates to a planimetric map providing no three-dimensional data about the terrain. Vectorization of layer pertaining to contours has a tremendous application to geo scientific modelling due to the potential 3D model representation of the terrain. Contour lines are represented by curves with 2-4 pixels width when scanned at a resolution of 300dpi. The curves are sometimes labelled indicating the elevation values above the mean sea level.

2 Motivation

The objective of an automatic map interpretation system is to generate a symbolic description for all map entities and their spatial relationships, which is then stored in a geographic database. Pattern recognition in cartographic documents aims at the delineation and extraction of spatial information and its incorporation into GIS as raster or vector data. However to establish methods for recognition in maps is particularly challenging because of the complexity of map contents as well as the presence of single and composite map elements which are heavily interconnected.

Topographic maps stored in paper form need to be converted into digital form for integration into geographic databases. Manual conversion of maps by an operator using a digitizing tablet, however, is time-consuming, costly, and error-prone. The development of a more efficient map input system is, therefore, of substantial importance. Topomaps belong to the class of documents and hence their analysis is considered as a branch of Document Image analysis. Therefore major steps to be followed include data capture, pixel level analysis and feature level analysis [KGG02]. Data capture is done with the use of a scanner as input device, and the

scanned image is subjected to further image analysis techniques, to automatically convert a paper-based map into atomic objects that can be interpreted and processed by a computer. However unlike many other documents where the focus is on separating text from graphics, the topomaps have to be processed to separate various semantic overlapping layers which are graphical in nature. Extraction of semantic layers, raster to vector conversion, recognition of symbols and text, 3D modelling etc. are some of the research challenges involved in topomaps processing which demand application of advanced image processing and vectorization algorithms. The following issues motivate research in the area of automatic topomap processing:

- Vectorization of Paper-based topographic maps is tedious and time-consuming process as it involves lot of manual effort.
- Not many of the existing techniques are focusing on color line drawings with thin closely spaced linear features.
- Current techniques of fully automatic vectorization for topographic maps are not efficient in extracting contour lines, due to the inherent features of topomaps described below.
- Processing large topomaps require enormously high execution time. Parallel programming has to be considered to obtain viable solution in reasonable time.

Several features in topomaps present formidable challenges to their processing:

- Aliasing and Existence of False colours : Introduced due to scanning process. For example in a simple scanned USGS map consisting only of blue, brown, black and purple, over 42000 distinct colours can be detected [PT08].
- Closely spaced features: When two features are closely spaced, the background separating them is eroded by the scanner induced aliasing to the point where it is difficult to use background to split these features.

- Intersecting and overlapping linear features: Topographic map is cluttered with data like contour lines, trails, streams, vertical coordinate grid line, etc. Separating each of these features can cause one or both of them to appear broken.

- Textured Background.

An approach which is fully automatic and less computationally intensive should be developed to extract the contour layer and to eliminate undesired features. In addition algorithms to detect and fill the discontinuities in contour lines caused due to overlapping/intersecting features of topomap, or improper scanning/extraction process have to be proposed. Since the goal of topomap processing is generation of a 3D model, and its incorporation into a GIS, a suitable representation scheme for contour layer is essential which can efficiently handle spatial operations at a region level and pixel level.

3 Literature Survey

Research in generation of contour layer of a topomap focuses mainly on extracting contour lines using various colour based segmentation methods, reconstructing contour layer and representation. When a contour layer is extracted from a colour topomap, two problems occur. Firstly, the contour layer has added noise/artifacts due to improper segmentation in addition to altitude values and hence a suitable filtering process to clean the artifcats and segment the text from contours is essential. Secondly, the contour lines have gaps and discontinuities due to separation of intersecting and overlapping layers. Therefore a reconstruction strategy is necessary to identify and fill these gaps. We present briefly survey on the following:

(a) Extraction of Contour Layer

(b) Representation of Contour Layer

(c) Reconstruction of Contour Layer

a) Extraction of Contour layer

Early segmentation methods on topomaps concentrated on color space selection. Ansoult et al. [ASL90] use the mean and variance of the hue channel for discriminating soil types on a digitized soil map. Ebi et al. [ELA94] transform the input RGB color space into another color space considering the chromaticity. Khotanzad et al. [KZ03] have addressed the problem by proposing a method to build a color key set for linear and area features of standard USGS topomaps and a non parametric clustering algorithm based on multidimensional histogram of the topomap. Yang Chen et al. [CWQ06] have extended this work beyond the standard USGS maps to common conditioned maps by using the features extracted from the gray version of the same color map along with the color coded map. Carol Rids et al. [RRA04] have used color edge detection - vector angle edge detector, with a saturation-based combination of hue and intensity planes, and color clustering (minimum variance quantization) to extract the contour lines from color coded maps. The clustered contour layer is subjected to further processing to separate the altitude tags coupled to the contour lines. Hence survey in the area of automatic separation of text and symbols from document images is included. Fletcher and Kasturi [FK88] have used connected components and some properties like Area etc. Morphology was used by Luo et al. [HGI95]. Cao et al. [RC01] have proposed a method to extract text touching to the graphics, but their method is done on a vectorized image. Tombre et al. [KSL$^+$02] have proposed a method based on connected components. P.P.Roy et al. [PVL$^+$08] have developed a system to segment text using color, and further by using geometric features of connected components and skeleton information in each layer.

b)Curve Representation

Various shape representation techniques can be classified into contour (boundary) based approaches and region based approaches. The classification is based on whether features are extracted from the curve (boundary) only or are extracted from the whole region. Under each class, different methods are further divided into structural approaches and global approaches. This sub-classification is based on whether the shape is represented as a whole or represented by segments/sections (primitives). Zhang and Lu [ZL04] have reported various methods with respect to this classification. Some of these methods use a hierarchical representation to build approximations of curves at several levels-of-detail or resolutions.

Contour based approaches

In Polygonal approximation, a set of vertices are located on the given curve such that when they are joined by straight lines, a polygon that captures the shape of the curve is formed. This can be achieved by dominant point detection [IJ84], [CR88], [BK92], [D.S93], [P.C97] etc. or by a search process [JE94], [P.Y99, P.Y03], [SY99], [M.S02] etc.

Hierarchical representation

Some standard hierarchical representations of curves are Arc tree [GW90], Strip tree [Bal81] etc. With increasing levels of hierarchy, the given curve is recursively subdivided into shorter sub curves, with straight line segments serving as approximations for each of these sub curves. The exact locations of the vertices (i.e., points of subdivision) are decided by the particular algorithm used for the purpose. A Strip tree recursively divides a curve segment into two parts at the point that has maximum error from the approximated straight line segment for the curve segment. Arc tree splits a curve into $2k$ segments of equal lengths at k-th hierarchical level. To achieve better performance than Arc tree, Sarkar et al. [SRS03] have introduced a new data structure called Equal error tree. The

idea behind this scheme is to select that particular point (on the curve) as the next point of subdivision for which the difference in errors for two sub-segments is either zero or minimum. Equal error tree outperforms Arc tree and compares well with Strip tree. Ganguly [Gan06] has modified Arc tree to overcome the data compression related shortcoming. Equal error tree has high computational complexity, best case $O(n^2 logn)$ and worst case $O(n^3)$, which may limit its usefulness for large scale curves.

Region based approaches

Common region based methods use moment descriptors to describe shapes. One such approach appears in [WY00] which uses Zernicke moments as a shape descriptor. Other region based methods include grid method, shape matrix, convex hull and median axis. Y.Ebrahim et al. [YMSA08] provided a new shape representation technique which utilizes the Hilbert space-filling curve. Armstrong et al. [AAC09] have expanded this method by providing a way to make the method rotation-invariant.

Hierarchical representation

Similar to the contour structural methods, region-based structural methods decompose the region into parts which are then used for shape representation and description. In the domain of hierarchical region representation, the Quad tree and Oct tree [GK79], [I.G82], [Sam84] have been proposed. Quadtree approach to region representation, termed a region quadtree, is based on the regular decomposition of the image into four equal-sized quadrants until each quadrant is entirely contained in the region or entirely disjoint from it.

c) Contour layer Reconstruction

Methods that have been developed focussing on reconstruction of contour lines can be classified as either Image based [LKH95], [AS99], [KZ03], [CWQ06] / Geomet-

ric based [NMD98], [KMA99], [SG03], which use either Local/Global approach and which use external information or not. In the image based approach, two segments are grouped based on the perceptual principles like proximity and continuity. The basic limitation of image based methods is the lack of utilization of domain knowledge e.g. contour lines do not intersect. Arrighi et al. [AS99] used Euclidean distances between extremities and weighted directions and combined both to compute a new distance for connecting the broken lines. Once all possible distances are computed, the extremities in lower distance are connected. The procedure is repeated until no extremities are left. This approach is also used in [KZ03] employing A* search algorithm. Geometric based approaches, on the other hand, transform the problem of raster-to-vector conversion into curve reconstruction. Spinello Salvatore et al. [SG03] vectorized contour line using Delaunay Triangulation where Delaunay edges are filtered using local and global rules. Du Jinyang et al [JY04] used principles of mathematical morphology (dilation) to acquire spatial relationships of contour lines for matching and connecting broken contour lines. Lastly, gradient vector flow approaches employ orientation field generated by the contour lines for reconstruction [ARFC06], [JS07]. An application to snakes is given in [XZZ06].

4 Contributions

Figure 2 is our visualization of various stages and the information flow in topomap processing. Though work has been done in the area of topographic map processing, research work which presents a systemic approach to the problem taking a colour topographic map as input, and giving a reconstructed contour layer along with a representation scheme as output has not been reported. Our thesis contribution is in the areas of Extraction, Contour line Representation and Reconstruction and is summarized in Figure 3.

Stage I: As mentioned earlier, a topomap consists of various features differen-

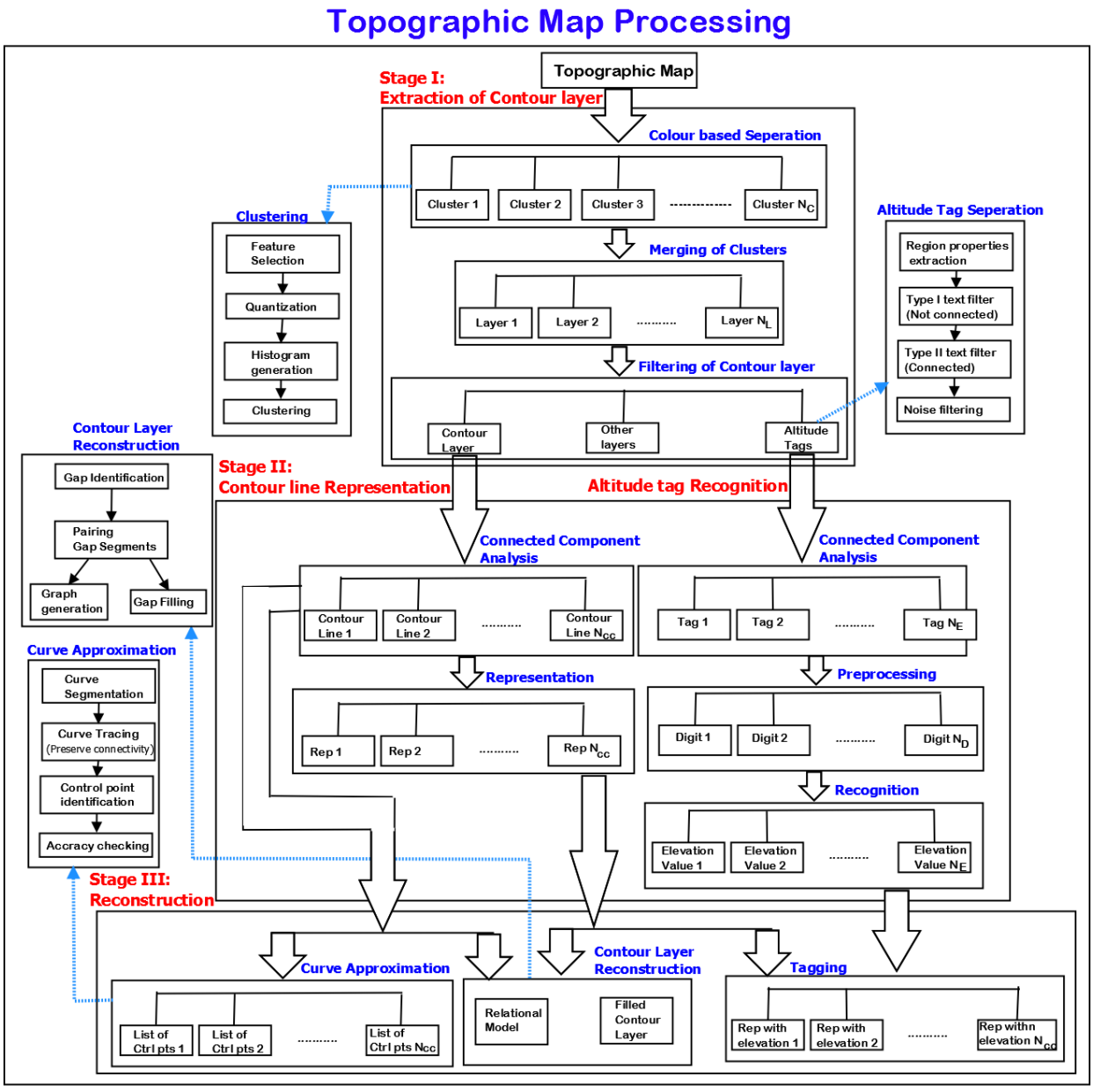


Figure 2: Topographic Map Processing

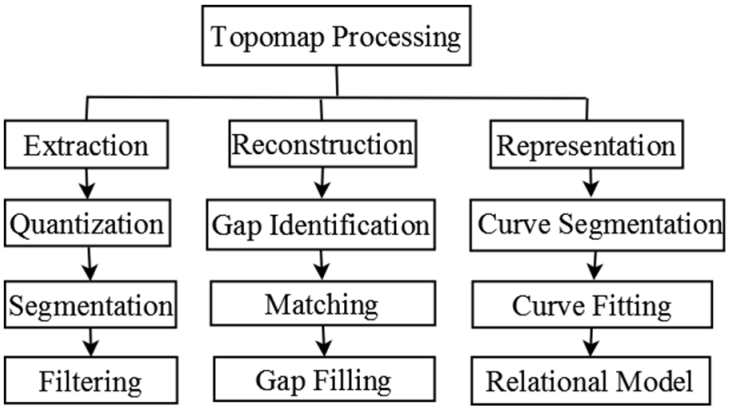


Figure 3: Stages of Topomap Processing

tiated primarily based on color. Hence clustering algorithms based on colour are applied on topomap to extract each of these features into a separate layer. The objective of the Stage I is to separate the contour layer from various other features of topomaps like water bodies,vegetation etc. Prior to applying clustering on a topomap, the colours are quantized such that the original colours of the image are mapped to a smaller subset of colours. Clustering is followed by merging of the original clusters which further helps in extraction of meaningful layers from the topomap. The topomap is thus segmented into various layers based on their colour features, for example a layer of water bodies (in blue), a layer of text and grid lines (in black), a layer of contour lines (in brown) etc.

The resulting contour layer is not clean, and consists of artifacts i.e some pixels of other layers not belonging to contour lines, due to problems of aliasing and false colours in a colour topographic map. Apart from contour lines and artifacts, the contour layer also contains the altitude values which are represented using the same color as contour lines. Hence algorithms for filtering the artifacts and for separation of text from graphic images are essential.

Stage II: After the Stage I, a clean contour layer, consisting of segments of contour lines is extracted from the topomap. The filtered altitude tags form another layer and the rest of the information of topomap is stored in 'other layer' as it is useful in later processing steps. Each contour line from the isolated contour layer

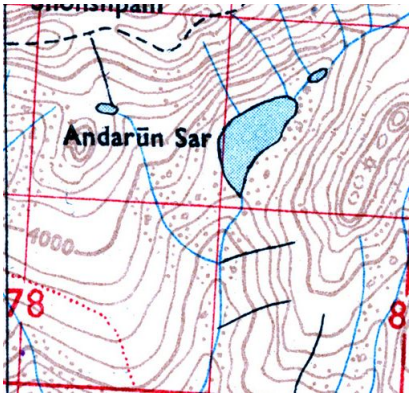


Figure 4: Input Topomap

is extracted and represented using curve representation techniques. Similarly altitude tags are segmented and recognized. This forms the second stage of topomap processing which outputs a contour line representation and recognized altitude tags.

Stage III: Extraction stage leads to gaps in contour lines due to intersecting and overlapping features. So prior to vectorizing the contour layer, it becomes essential to fill the gaps. The third stage is Reconstruction, which involves the algorithms for identifying contours which contribute to a gap, pairing contour segments which form a gap and then filling the gaps. This stage takes a binary contour image, which is the output of Extraction stage as input and generates a reconstructed binary contour image as output. The recognized elevation values are tagged to the contour lines and an attribute which stores the altitude value is updated in the representation of contour line.

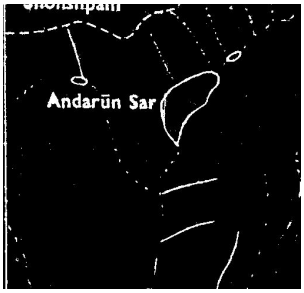
We now briefly discuss our main contributions made to realize the above three stages. Results are shown using the topomap shown in Figure 4.

1. **Contour layer Extraction**: Binary image of contour layer is obtained from a topomap by employing color clustering algorithms. The first step of topomap segmentation has been performed using a modified non parametric histogram based clustering algorithm proposed by Khotanzad et al. [KZ03].

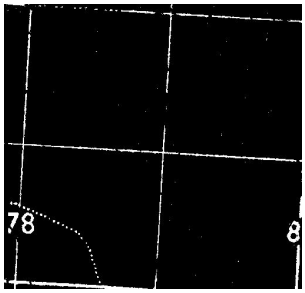
The clustering is performed with colour features extracted through a feature selection approach. The resulting clusters are post processed using neighbourhood analysis to obtain meaningful layers of topomap. Figure 4 shows the input image to the stage and Figure 5 shows the output of this stage, which are the various layers of topomap.

2. **Filtering of Contour layer:** The resultant contour layer contains altitude values in addition to the contours, as they are represented using the same color as contours. The filtering method proposed helps to produce a clean contour layer free from elevation values, which are connected to the contour lines. Filtering is done in two steps. Clustering has been used to separate the text associated with contours in the first step. The results from first step are used in the second step to detect text connected to contours. The extracted text can be recognized and tagged to the contour line, to generate a 3D model. Hence recognition of altitude tags which is an important precursor for generation of 3D model from the contour layer, is attempted using convolutional neural network, LeNet - 5 [YLYP98]. Figure 7(a) shows the contour layer after filtering text and artifacts from the image shown in Figure 5(d)

3. **Representation of contour layer:** A hierarchical scheme of representing the contour layer of a topomap is proposed which can handle various operations on the contour lines at a region level or at a pixel level. The representation scheme is based on the threaded binary tree data structure, created from curve segmentation of contours, which offers several benefits like easy retrieval of information and efficient storage. Contours are segmented based on the density ratio and are further approximated to a cubic Bezier curve, which facilitates visualisation. The proposed binary threaded tree representation scheme also aids in identifying the gaps in contours, which can subsequently be filled. Figure 6(a) shows the input image, and Figure

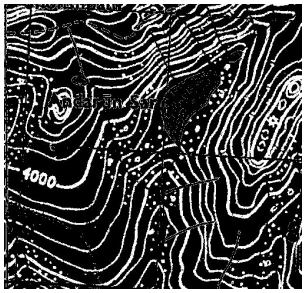


(a) Text & Boundaries(Black)



(b) Grid Lines(Red)

(c) Streams(Blue)

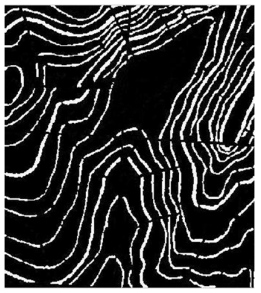


(d) Contours(Brown)

Figure 5: Output of Stage I

6(b) shows the output, where the segments of each contour are shown by red marks and green curves indicate Bezier approximation of each segment.

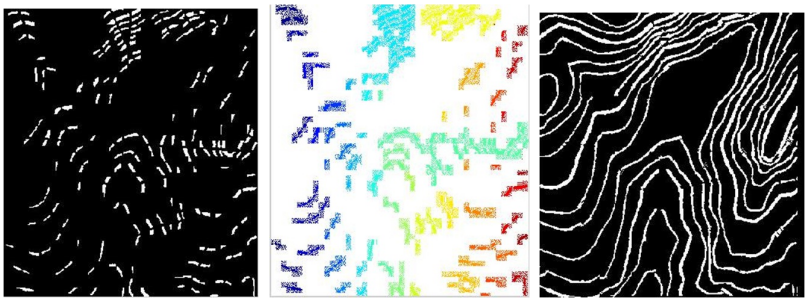
4. **Reconstruction of contour layer:** A novel hybridized algorithm is developed for reconstructing the extracted contour lines from color topographic map. This has been addressed by developing algorithms based on connected components, graph theory, Expectation Maximization (EM) and numerical



(a) Input Image

(b) Output Image

Figure 6: Output of Stage II



(a) Gap Contour Image (b) Localization (c) Filled Image

Figure 7: Output of Stage III

methods. The algorithm operates by isolating the segments of those contours which have gaps using an overlapping sliding window analysis of binary contour image. It achieves reduction in the complexity of the matching by employing the EM algorithm on the image obtained from Gap identification. The scalability of Gap identification algorithm is tested by implementing it on a parallel architecture. A new scheme of filling gaps assumes thick contours without the need of thinning it. Figure 7 shows the output images of this stage, with Figure 6(a) as input. Figure 7(a) is the output after applying Gap Identification algorithm, Figure 7(b) the image after applying Localization to Gap Contour Image and Figure 7(c) shows the image after applying Weaving between two matched gap segments.

We present results of our work with those reported in literature.

4.1 Publications

Following is the list of publications:

1. Abstract on " Topographic map processing on GEON ", Proceedings of AP Science Congress, 2008
2. " Automatic gap identification towards efficient contour line reconstruction in topographic maps." Asia International Conference on Modelling and Simulation, p:309 - 314, 2009

3. " Adaptive Altitude Feature Recognition for Paper based Topographic map ", The 4th Mahasarakham International Workshop on Artificial Intelligence, Thailand, 2010

4. " Representation of Contour lines Extracted from Topographic map ", communicated to Computers & Geosciences - Elsevier

5 Organization of the thesis

Our work is organized into seven chapters as follows:

Chapter 1. introduces topographic maps and their application in the area of Geo Science and motivation for pursuing research in this area.

Chapter 2. reviews the research done in the various areas of topomap processing.

Chapter 3. deals with the Extraction of contour layer which includes Feature selection, Clustering, Separation and Recognition of altitude tags.

Chapter 4. proposes a new scheme for representation of the contour layer and briefly discusses how spatial queries are handled.

Chapter 5. explains the proposed Gap identification i.e identifying the broken segments of contour line and matching the segments which should be joined to form a closed contour. The chapter also includes implementation details of Gap Identification algorithm on a parallel architecture.

Chapter 6. focuses on the various pre processing operations for reconstructing a surface from the contour layer like Approximating contour segments, Gap filling and tagging recognized altitude values to the contour lines.

Chapter 7. concludes the thesis by consolidating the contributions made, showing results of the proposed algorithms and discusses the future scope of the research in the area.

6 Conclusion

A survey of the methods for extraction of contour lines from topomaps has been done. A colour topographic map is clustered into layers like contour, vegetation, water bodies etc. based on colour features. However the contour layer is subjected to further processing as the altitude tags which represent the elevation value of the contour are also present along with the lines. A scheme of filtering altitude tags attached to the contours is discussed. It can be observed that the labels thus extracted can be further useful to tag the contour lines with the altitude values for generating a 3D model. All the steps in the proposed approach are highly amenable to complex shapes of contour lines.

A representation for contour layer of topomap has been proposed. Gap identification and filling which are essential pre-operations of contour layer vectorization have been performed efficiently using the proposed representation. In addition the representation is able to handle operations necessary for visualization like affine transformations and multi resolution viewing of the curves.

Further the proposed method includes an algorithm for identification of contour segment end points, thereby, making the entire process of contour line reconstruction fully automated. The algorithm integrates a scheme of identifying contour line segments in the vicinity of the gap which is an aid to tracing algorithms. A new technique using EM algorithm to reduce the pairs of segments to be matched has been proposed in our approach leading to significant reduction in the complexity of matching process. The recursive method for gap identification can be applied not only to topomaps but also to a variety of documents involving line drawings. The algorithm for filling of gaps between contour line segments of any thickness and orientation has been introduced.

Future work includes developing algorithms for other layers of topomap and designing a knowledge based system for topomap digitization, building 3D model from digitized contour layer. Also, possibility of extending proposed algorithm of

extraction of layers to any colour document will be explored. Grid based implementation of the algorithms can be an interesting and challenging task.

References

[AAC09] Jeffrey Armstrong, Maher Ahmed, and Siu-Cheung Chau. 'A Rotation-Invariant Approach to 2 D Shape Representation Using the Hilbert Curve'. *ICIAR 2009, LNCS*, 5627:594–603, 2009.

[AK05] C.P.Lo Albert and K.W.Yeung. *Concepts and techniques of Geographic Information Systems*. PH Series in Geographic Information Science, Prentice-Hall, India, 2005.

[ARFC06] A.Chessel, R.Fablet, F.Cao, and C.Kervrann. Orientation interpolation and applications. *Proceedings of IEEE International Conference on Image Processing:*, pages 1561 – 1564, 2006.

[AS99] Patrice Arrighi and Pierre Soille. From scanned topographic maps to digital elevation models. *Proceedings of Geovision99, International Symposium on Imaging Applications in Geology*, 1999.

[ASL90] M. Ansoult, P. Soille, and J. Loodts. 'Mathematical morphology: A tool for automated GIS data acquisition from scanned thematic maps'. *Photogramm.Eng. Remote Sens*, 56(9):1263 1271, 1990.

[Bal81] Dana H. Ballard. Strip. trees:a hierarchical representation for curves. *Graphics and Image Processing, Communications of the ACM*, 24(5):310 – 321, 1981.

[BK92] B.K.Ray and K.S.Ray. An algorithm for detection of dominant points and polygonal approximation of digitized curves. *Pattern Recognition Letters*, 13:849–856, 1992.

[CR88] C.H.Teh and R.T.Chin. On image analysis by the methods of moments. *IEEE Trans. Pattern Anal. Mach. Intell PAMI*, 10(4):496–513, 1988.

[CWQ06] Yang Chen, Runsheng Wang, and Jing Qian. Extracting contour lines from common-conditioned topographic maps. *IEEE Transactions on Geoscience and Remote Sensing*, 44(4):1048 – 1057, April 2006.

[D89] Rhind D. 'Why GIS?'. *ARC News*, 11(3), 1989. CA: Environmental Systems Research Institute,Inc.

[D.S93] D.Sarkar. A simple algorithm for detection of significant vertices for polygonal approximation of chain-coded curves. *Pattern Recognition Letters*, 14:959–964, 1993.

[ELA94] N. Ebi, B. Lauterbach, and W. Anheier. An image analysis system for automatic data acquisition from colored scanned maps. *Machine Vision Applications*, 7(3):148 164, 1994.

[FK88] L.A. Fletcher and R. Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Transactions on pattern Analysis and Machine Intelligence*, 10:910 – 918, 1988.

[Gan06] Pankaj Ganguly. Modified arc tree based hierarchical representation of digital curve. *Pattern Recognition Letters*, 27:529–535, 2006.

[GK79] G.M.Hunter and K.Steiglitz. Operations on images using quadtrees. *IEEE Transactions on PAMI*, 1(2):145–153, 1979.

[GW90] Oliver Gunther and Eugene Wong. The arc tree :an approximation scheme to represent arbitrary curved shapes. *Comput.Vision Graphics Image Process.*, 51:313–337, 1990.

[HGI95] Luo H.Z, Agam G, and Dinstein I. Directional mathematical morphology approach for line thinning and extraction of character strings from maps and line drawings. *International Conference on Document Analysis and Recognition(ICDAR)*, 1:257, 1995.

[I.G82] I.Gargantini. An effective way to represent quadtrees. *Comm. ACM*, 25(12):905–910, 1982.

[IJ84] I.M.Anderson and J.C.Bezdek. Curvature and tangential deflection of discrete arcs: a theory based on the commutator of scatter matrix pairs and its application to vertex detection in planar shape data. *IEEE Trans Pattern Anal.Machine Intell*, 6:27–40, 1984.

[JE94] J.C.Perez and E.Vidal. Optimal polygonal approximation of digitized curves. *Pattern Recognition Letters*, 15:743–750, 1994.

[JS07] J.Pouderoux and S.Spinello. Global contour lines reconstruction in topographic maps. *Ninth International Conference on Document Analysis and Recognition*, 2:779–783, 2007.

[JY04] Du Jinyang and Zhang Yumei. Automatic extraction of contour lines from scanned topographic map. *IEEE*, pages 2886–2888, 2004.

[KGG02] Rangachar Kasturi, Lawrence O Gorman, and Venu Govindaraju. Document image analysis: A primer. *Sadhana*, 27(1):3 – 22, February 2002.

[KMA99] Tamal K.Dey, Kurt Mehlhorn, and Edgar A.Ramos. Curve reconstruction: Connecting dots with good reason. *Proceedings of the 15th Symposium on Computational Geometry*, 15:197 206, 1999.

[KSL+02] Tombre K, Tabbone S, Peissier L, Lamiroy B, and Dosch P. Text /graphics separation revisited. *In Document Analysis Systems V*, 2423:615 – 620, 2002.

[KZ03] Alireza Khotanzad and Edmund Zink. Contour line and geographic feature extraction from usgs color topographical paper maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):18 – 31, 2003.

[LKH95] L.Eikvil, K.Aas, , and H.Koren. Tools for interactive map conversion and vectorization. *Third International Conference on Document Analysis and Recognition: ICDAR*, 2:927 – 930, 1995.

[M.S02] M.Salotti. Optimal polygonal approximation of digitized curves using the sum of square deviation criterion. *Pattern Recognition*, 35:435–443, 2002.

[NMD98] N.Amenta, M.Bern, and D.Eppstein. The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical models and image processing*, 60(2):125 – 135, 1998.

[P.C97] P.Cornic. Another look at the dominant point detection of digitized curves. *Pattern Recognition Letters*, 18(4):13–25, 1997.

[PT08] Aria Pezeshk and Richard L. Tutwiler. Contour line recognition and extraction from scanned colour maps using dual quantization of the intensity image. *IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, pages 173 – 176, March 2008.

[PVL+08] P.P.Roy, Eduard Vazquez, Josep Llados, Ramon Baldrich, and Umapada Pal. A system to segment text and symbols from color maps. *7th International Workshop on Graphics Recognition Recent Advances and New Opportunities: LNCS*, 5046:245 – 256, 2008.

[P.Y99] P.Y.Yin. Genetic algorithms for polygonal approximation of digital curves. *Int. J. Pattern Recognition Artificial Intell*, 13:1061–1082, 1999.

[P.Y03] P.Y.Yin. Ant colony search algorithms for optimal polygonal approximation of plane curves. *Pattern Recognition*, 36:1783–1797, 2003.

[RC01] Cao R and Tan C.L. Text/graphics separation in maps. *Proceedings of 4th IAPR International Workshop on Graphics Recognition*, 1:44 48, September 2001.

[RRA04] Carol Rids, Corneliu Rusu, and Jaakko Astolu. Contours extraction from a colour coded relief scanned map. *International Conference on Image Processing (ICIP)*, 3:1699 – 1702, Oct 2004.

[Sam84] H. Samet. The quadtree and related hierarchical data structures. *ACM Comput. Surveys*, 16(2):187–260, 1984.

[SG03] Spinello Salvatore and Pierre Pascal Guitton. Contour line recognition from scanned topographic maps. *Journal of WSCG*, 12(1-3), February 2003.

[SRS03] Biswajit Sarkar, Sanghamitra Roy, and Debranjan Sarkar. Hierarchical representation of digitized curves through dominant point detection. *Pattern Recognition Letters*, 24:2869–2882, 2003.

[SY99] S.C.Huang and Y.N.Sun. Polygonal approximation using genetic algorithms. *Pattern Recognition*, 32:1409–1420, 1999.

[WY00] W.Y.Kim and Y.S.Kim. A region-based shape descriptor using zernike moments. *Signal Processing: Image Communication*, 16(1-2):95–102, 2000.

[XZZ06] Dongjun Xin, Xianzhong Zhou, and Huali Zheng. Contour line extraction from paper-based topographic maps. *Journal of Information and Computing Science*, 1(5):275–283, 2006.

[YLYP98] LeCun Y., Bottou L., Bengio Y., and Haffner P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278 – 2344, November 1998.

[YMSA08] Y.Ebrahim, M.Ahmed, S.C.Chau, and W. Abdelsalem. Shape representation and description using the hilbert curve. *Pattern Recognition Letters*, 30(4):345–358, 2008.

[ZL04] Dengsheng Zhang and Guojun Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37:1–19, 2004.