AUTOMATIC OBJECT EXTRACTION IN A MULTISPECTRAL IMAGES

SARANYA DEVI.D
10PCA21

A Project Report submitted to Avinashilingam Deemed University for Women, Coimbatore in partial fulfilment of the requirements for the Master's Degree in Computer Applications

MAY 2013

AUTOMATIC OBJECT EXTRACTION IN A MULTISPECTRAL IMAGES

SARANYA DEVI.D
10PCA21

A Project Report submitted to Avinashilingam Deemed University for Women, Coimbatore in partial fulfilment of the requirements for the Master's Degree in Computer Applications

MAY 2013

SIGNATURE OF THE HEAD OF THE DEPARTMENT

SIGNATURE OF THE SUPERVISOR

SIGNATURE OF THE EXTERNAL EXAMINER

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

I would like to express my sincere thanks to God Almighty, for His constant love and grace that He has showered upon me.

I am very grateful to **Dr.T.S.K.Meenakshi Sundaram, M.A., M.Phil., Ph.D., Chancellor,** Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore for his support and encouragement during the course of my study.

I heartily thank **Dr. (Mrs).Sheela Ramachandran M.Sc., P.G. Dip., Ph.D., Vice Chancellor,** Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore for extending all resources that facilitated the conduct of the present study.

I express my humble gratitude to **Dr. (Mrs). Gowri Ramakrishnan M.Sc., M.Phil., Ph.D., Registrar,** Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing all facilities necessary for the study.

I am also thankful to **Dr. (Mrs). R. Parvatham M.Sc., Dip.Ed. M.Phil., Ph.D., Dean Faculty of Science,** for granting the facility required.

I wish to place on record my deep sense of gratitude to **Dr. (Mrs).G.Padmavathi M.Sc., M.Phil., Ph.D.,** Professor and Head, Department of Computer Science, for providing all the facilities to complete the project.

I owe great deal of gratitude to my esteemed guide **Mrs.N.VALLIAMMAL M.SC, M.Phil, Assistant Professor,** Department of Computer Science, for imparting the tremendous assistance and well timed support for triumph of my project.

I take this unique opportunity to express my sincere thanks to my project Coordinator **Dr.(Mrs)V.SRIVIDHYA, M.SC., M.Phil., Ph.D., Assistant Professor,** Department of Computer Science, for her kind advice and knowledgeable suggestion, which helped me to complete my project successfully.

I would extend my hearty thanks to one and all who helped me directly or indirectly for successful completion of my project.

SYNOPSIS

SYNOPSIS

To propose a new automatic image segmentation method.Color edges in an image are first obtained automatically by combining an improved isotropic edge detector and a fast entropic thresolding technique .After the obatained color edges have provided the major geometric structures in an image ,the centroids between these adjacent edge regions are taken as initial seeded region growing(SRG).These seeds are then replaced by the centroidos of the generated homogenous image regions by incorporating the required additional pixels step by step.morever ,the results of color edge extraction and SRG are integrated to provide homogeneous image region with accurate and closed boundaries.We also discuss the application of our image segementation method to automatic face detection .furthermore,semantic human objects are generated by a seeded region aggregation procedures which taken the detected faces as object seeds

This schemes utilizes a predictive coding model to identify the direction of change in color and texture at each location at agiven scale ,and constructs an edge flow vector .By propagating the edge flow vectors ,the boundaries can be detected at image locations which encounters two opposite direction of flow in the stable state .A user defined image scale is the only significant control parameter that is needed by the algorithm .The scheme facilitiates integration of color and texture into single framework for boundary detection .Segemntation results on a large and diverse collection of natural images are provided ,demonstrating the usefulness of this method to content based image retrievel

CONTENTS

CONTENTS

S.NO	PARTICULARS	Page No
1.	**INTRODUCTION**	
	1.1 PROBLEM DESCRIPTION	1
2.	**SYSTEM CONFIGURATION**	
	2.1 HARDWARE SPECIFICATION	6
	2.2 SOFTWARE SPECIFICATION	7
	2.3 ABOUT THE SOFTWARE	7
3.	**LITERATURE SURVEY**	
	3.1 EDGE DETECTION	8
	3.2 THRESHOLD	17
	3.3 CONTOUR LINKING	18
	3.4 MATHEMATICAL MORPHOLOGY	18
	3.5 CONTOUR LABELING	19
	3.6 CONTOUR FILLING	21
	3.7 STUDYING BMP FILE FORMAT	22
4.	**SYSTEM STUDY AND REQUIREMENT SPECIFICATION**	
	4.1 PROBLEM FORMULATION	23
	4.2 SYSTEM STUDY	23
	4.3 REQUIREMENT SPECIFICATION	29
	4.4 EXISTING SYSTEM	31
	4.5 PROPOSED SYSTEM	32
5.	**ARCHITECTURE DESIGN**	
	5.1 OVERALL SYSTEM DESIGN	33
	5.2 I/O INTERFACE DESIGN	34

6. IMPLEMENTATION METHODOLOGY

6.1 IMPLEMENTATION STAGES 35

6.2 MODULES 36

7. CODING AND TESTING

7.1 CODING RULE 38

7.2 TESTING 40

8. CONCLUSION AND FUTURE IMPROVEMENT 41

BIBLIOGRAPHY 42

APPENDIX

SCREEN SHOTS 43

INTRODUCTION

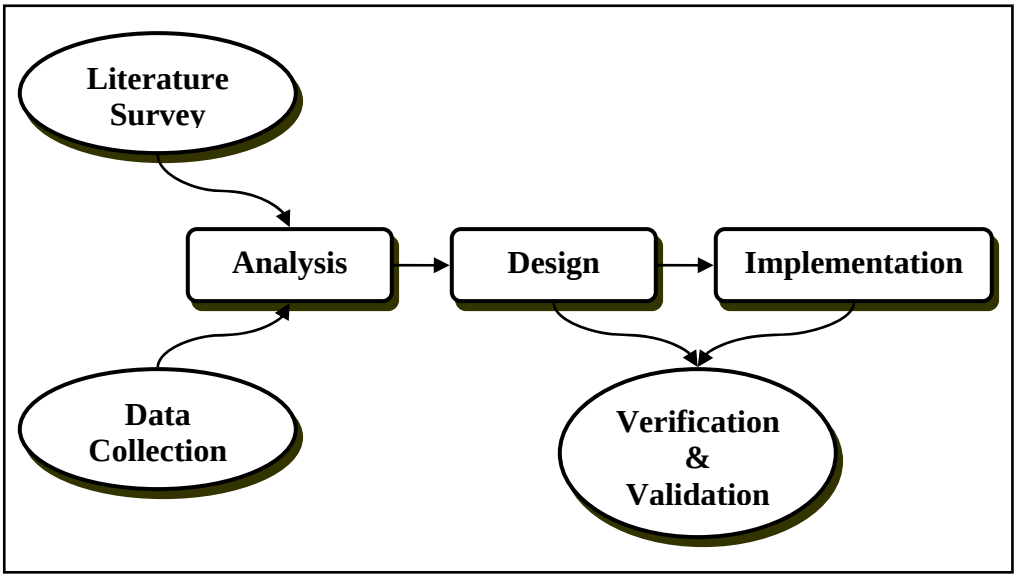
INTRODUCTION

1.1 GENERAL OVERVIEW

In most computer vision applications, image segmentation constitutes a crucial initial step before performing high-level tasks such as object recognition and scene interpretation. While considerable research and progress have been made in the area of image segmentation, the robustness and generality of the algorithms on a large variety of image data have not been established. One of the major difficulties arises from the fact that most natural images are rich in color and texture, and these features need to be integrated for a good segmentation. Furthermore, image segmentation itself is an ill-posed problem.

In the past few years there has been a tremendous growth in the multimedia computing technology. It now pervades many aspects of day-to-day life, including television and broadcasting, medicine, engineering, and life sciences. A sophisticated technology exists to generate, store, and transmit large amounts of image, video, and audio data. The day is not too far when every piece of information that one needs can be obtained at the stroke of a data will be of interactive nature. For example, the new MPEG4 video compression standard allows content/object based coding for interactive video applications. However, in order for this to happen, robust algorithms for image/video content analysis need to be developed. The work described in the following is concerned with the development of new image/video segmentation algorithms that are suitable for large databases and require very little parameter tuning.

The overall project flow is depicted in the following fig 1.1



Object detection in images can be separated into two levels, low-level object detection and high-level object detection, according to the level of target object. The low-level object detection, also known as image segmentation is usually based on the analysis of color similarity. In the following, we describe the segmentation procedure

Image segmentation procedures can be broadly divided into three classes, namely,

- Pixel-oriented,
- Region-oriented, and
- Edge-oriented algorithms.

Pixel and region-oriented schemes segment the image based on similarity of pixel attributes within regions while edge-oriented segmentation, based on discontinuity between regions, focuses on extracting edges or boundaries rather than regions.

Edge-based segmentation represents a large group of methods based on information about edges in the image; it is one of the earliest segmentation approaches and still remains very important. We develop an improved isotropic color-edge detector by including more potential edge patterns and integrating a fast entrophic thersholding technique.

Edge in digital color image

Edge detection is a problem of fundamental importance in image analysis. In typical images, edges characterize object boundaries and are therefore useful for segmentation, registration, and identification of objects in a scene. An edge is a jump in intensity. The cross section of an edge has the shape of a ramp. An ideal edge is a discontinuity (*i.e.*, a ramp with an infinite slope).

A first intuitive answer is that the respective brightness values of two neighboring pixels in an edge should be significantly or steadily different. An alternative edge characteristic is that the colors of the neighboring pixels are significantly different, even though their brightness values are very similar. Therefore, both changes in the brightness and changes in the color between neighboring pixels should be exploited for more efficient color-edge extraction. We choose instead to implement a simpler, though less accurate isotropic edge detector for identifying the geometric structures of an image.

There are two categories of isotropic edge detectors:

1. Gradient operators and

2. Second derivative operators.

Gradient operators, such as the Roberts, Prewitt, and Sobel operators, detect the edges by looking for the maximum and minimum in the first derivative of the luminance

The second derivative operators, such as the Marr-Hildreth and Marr-Poggio operators, search for zero-crossings in the second derivative of the luminance of an image to find the edges.

CONTOUR LINKING

Contour linking (also called boundary detection, contour detection) can be classified as a mid level image recognition process. Why a mid level? It lies between low-level techniques like edge detection and higher-level image processing like image segmentation or object recognition for example. Edge detection will choose edge pixels based on factors like color or brightness through computing their gradient magnitude and direction. However, this output is cannot be changed higher-level image processing. It is necessary to group individual pixels into continuous contours and link contours that have gaps. Noise or weak features of an edge can cause these discontinuities. Therefore some further processing is necessary.

Therefore the goal of edge linker is to find a continuous path through points approximated by object boundary.

THRESHOLDING

Thresholding represents the simplest image segmentation process, and it is computationally inexpensive and fast. A brightness constant called a threshold is used to segment objects and background.

Single threshold can either be applied to the entire image (global threshold) or can vary in image parts (local Threshold). Threshold detection methods are used to determine the threshold automatically.

Optimal thresholding determines the threshold as the closest gray-level corresponding to the minimum probability between the maxima of two or more normal distributions. Such thresholding results in minimum error segmentation.

MORPHOLOGICAL OPERATIONS

The word morphology commonly denotes a branch of biology that deals with the form and structure of animals and plants. We use the same word here in the context of mathematical morphology as a tool for extracting image components that are useful in the representation and description of region shape, such as boundaries, skeleton, etc.

OBJECT LABELING

The result of any successful segmented image is the labeling of each pixel that lies within a specific distinct segment. One means of labeling is to append to each pixel of an image the label number or index of its segment. A more succinct method is to specify the closed contour of each segment. A segment or region is a group of pixels of the same color, any two of which can be connected by continuous path of neighbors belonging to the group.

1.2 SCOPE OF THE PROJECT

- ✓ Partitioning into meaningful regions, i.e. Objects
- ✓ Feature extraction of object shape, optical density, and texture
- ✓ Surface visualization, image registration and compression
- ✓ Multimedia Internet search engine based on Content based retrieval

SYSTEM CONFIGURATION

2. SYSTEM CONFIGURATION

This section describes the hardware and software specification needed for both development and implementation phases of this project.

2.1 HARDWARE SPECIFICATION

Processor	: Intel® Pentium®
RAM	: 512 MB
Monitor	: 14 Inch HCL Color Monitor
Keyboard	: HCL
Pointing device	: Optical Mouse

2.2 SOFTWARE SPECIFICATION

Tool : VC++

Operating System : Microsoft Windows XP

2.3 ABOUT THE SOFTWARE

An application development tool developed by MICROSOFT for C++ programmer .Visual C++ supports object oriented programming of 32 bit Windows application with an integrated development environment(IDE),a C/C++ complier, and a class library called the Microsoft Foundation Classes(MFC),The IDE includes an AppWizard ,ClassWizard,and testing features to make programming easier

It provides a way of encoding name and additional information about a function ,structure ,class or another datatype in order to pass more semantic information from the Microsoft C++ complier to its linker .Visual Studio and the Windows SDK come with the program und name which may be invoked to obtain the C-style function prototype encoded in a mangled name .the information name. The information below has been mostly reverse – engineered .There is no official documentation for the actual algorithm used

3 LITERATURE SURVEY

3.1 EDGE DETECTION

Edges are very important to any vision system (biological)

- They are fairly cheap to compute
- They do provide strong visual clue that can help the recognition process.

Two major classes of differential edge detection

1. First order derivative

Some form of spatial first order differentiation is performed, and the resulting gradient is compared to a threshold value. An edge is present if the gradient exceeds the threshold.

2. Second order derivative

An edge is present if there is a significant spatial change in the polarity of the second derivative.

An edge may be regarded as a boundary between two dissimilar regions in an image. These may be different surfaces of the object, or perhaps a boundary between light and shadow falling on a single surface. In principle an edge is easy to find since differences in pixel values between regions are relatively easy to calculate by considering gradients.

- Extracting Edges from Images
- Detecting Edge Points
    - Gradient based method
    - Second Order Methods

3.1.1 EDGE DETECTION CONCEPTS

Edge detection is a hill defined term, because it makes think that the algorithm gives contours as result. In fact these algorithms give images, which show higher intensity in pixel near gray value transitions. In some way, this task is only ***contour enhancement***. Our conclusion from this experience is that real edge detection must produce as output: vector data representing contours. Any other results are only pretty images to make demos for visitors. In other word, the quality of an edge detection algorithm can only be evaluated objectively when it is used to extract contour represented in vector format (as a sequence of connected points).

The now widely accepted method of contour detection consists of a smoothing filtering followed by a derivative filtering. Smoothing is intended to reduce noise in the image without eliminating contours, usually a compromise should be found to the degree of smoothing with respect to the type of noise present in the image and the sharpness of contours. The derivative filter detects transitions or changes in gray levels in the image. Usually second derivative are used to determine precisely the location of maximum rate of change in gray level.

3.1.2 TYPES OF EDGE DETECTION

- Gradient Edge Detector
- Homogeneity Edge Detector

- Difference Edge Detector
- Variance Edge Detector

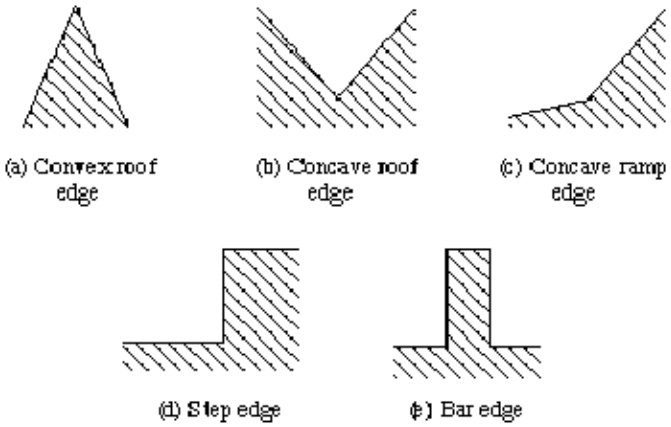
Many edge extraction techniques can be broken up into two distinct phases:

- Finding pixels in the image where edges are likely to occur by looking for discontinuities in gradients.
- Candidate points for edges in the image are usually referred to as ***edge points*** or ***edge pixels***
- Linking these edge points in some way to produce descriptions of edges in terms of lines, curves ***etc.***

GRADIENT BASED METHODS

The first derivative at any point in an image is obtained by using the magnitude of the gradient at that point. A change of the image function can be described by a gradient that points in the direction of the largest growth of the image function.

An edge point can be regarded as a point in an image where a discontinuity (in gradient) occurs across some line. A discontinuity may be classified as one of three types. This is shown in the fig 2.1



GRADIENT DISCONTINUITY

The gradient of the pixel values changes across a line. This type of discontinuity can be classed as

- ***roof*** edges
- ***ramp*** edges
- ***convex*** edges
- ***concave*** edges

Ramp edges have the same signs in the gradient components on either side of the discontinuity, while roof edges have opposite signs in the gradient components.

BAR DISCONTINUITY

Pixel values rapidly increase then decrease again (or ***vice versa***) across some line.

For example,

if the pixel values are depth values,

- ❖ jump discontinuities occur where one object occludes another (or another part of itself).
- ❖ Gradient discontinuities usually occur between adjacent faces of the same object.

GRADIENT

It is a vector, whose components measure how rapidly pixel values are changing with distance in the *x* and *y* directions.

Thus, the components of the gradient may be found using the following approximation:

$$\frac{\partial f(x,y)}{\partial x} \approx \Delta_x = \frac{f(x+d_x,y)-f(x,y)}{d_x},$$
$$\frac{\partial f(x,y)}{\partial y} \approx \Delta_y = \frac{f(x,y+d_y)-f(x,y)}{d_y},$$

where *dx and dy s*

In (discrete) images we can consider dx and dy in terms of numbers of pixels between two points. Thus, when **dx=dy=1**(pixel spacing) and we are at the point whose pixel coordinates are (*i*,*j*)

we have

d_y

$$\Delta_x = f(i+1,j)-f(i,j),$$
$$\Delta_y = f(i,j+1)-f(i,j).$$

$d_x = d_y = 1$

$d_x = d_y = 1$

In order to detect the presence of a gradient discontinuity we must calculate the ***change in gradient*** at (i,j). We can do this by

$$M = \sqrt{\Delta_x^2 + \Delta_y^2}$$

and the ***gradient direction***, given by

$$\theta = \tan^{-1}\left[\frac{\Delta_y}{\Delta_x}\right].$$

3.1.3 SAMPLE IMPLEMENTATION

The difference operators in correspond to convolving the image with the two masks in This is easy to compute:

- The top left-hand corner of the appropriate mask is superimposed over each pixel of the image in turn,
- A value is calculated for Δx *or* Δy by using the mask coefficients in a weighted sum of the value of pixel (i,j) and its neighbours.
- These masks are referred to as ***convolution masks*** or sometimes ***convolution kernels***.

-1	1
0	0

Δ_x

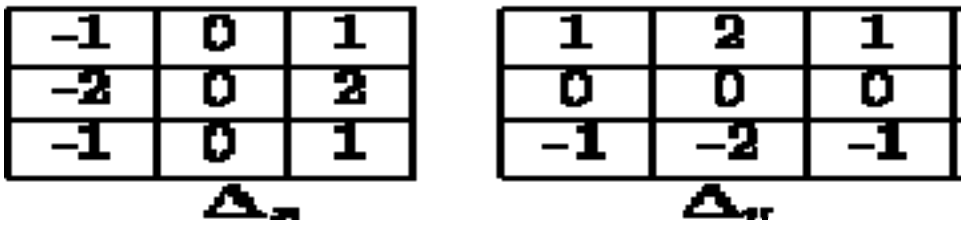
-1	0
1	0

Δ_y

Many edge detectors have been designed using convolution mask techniques, often using $3X3$ mask sizes or even larger. An advantage of using a larger mask size is that errors due to the effects of noise are reduced by local averaging within the neighbourhood of the mask. An

advantage of using a mask of odd size is that the operators are ***centred*** and can therefore provide an estimate that is biased towards a centre pixel (*i*,*j*).

One important edge operator of this type is the ***Sobel edge operator***.



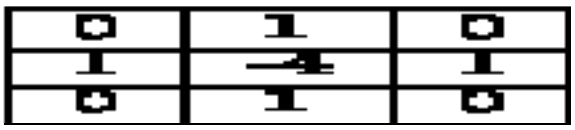
Sobel edge operator convolution masks

SECOND ORDER DERIVATIVE METHODS:

All of the previous edge detectors have approximated the first order derivatives of pixel values in an image. It is also possible to use ***second order derivatives*** to detect edges. A very popular second order operator is the ***Laplacian*** operator.

The Laplacian of a function *f*(*x*,*y*), denoted by $\nabla 2 f(x, y)$, is defined by:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}.$$



Laplacian operator convolution mask

Disadvantages

- Second derivatives will exaggerated noise twice as much.
- No directional information about the edge is given.

The problems that the presence of noise causes when using edge detectors means we should try to reduce the noise in an image prior to or in conjunction with the edge detection process. We have already discussed some methods of reducing or smoothing noise in the Image Processing Section. Some of these methods may be of use here.

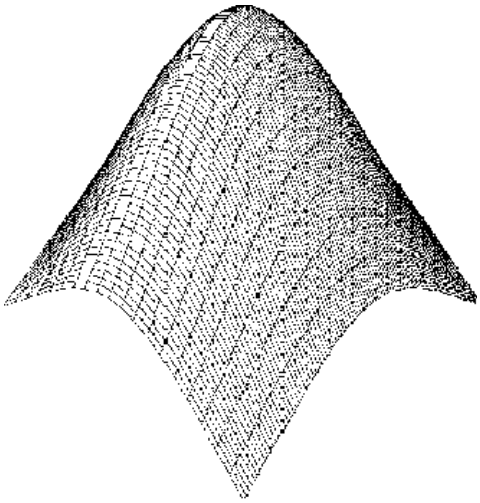
GAUSSIAN SMOOTHING

- Gaussian smoothing is performed by convolving an image with a Gaussian operator which is defined below.
- By using Gaussian smoothing in conjunction with the Laplacian operator, or another Gaussian operator, it is possible to detect edges.

The ***Gaussian distribution*** function in two variables, $g(x,y)$, is illustrated and is defined by

$$g(x, y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

Having smoothed the image with a Gaussian operator we can now take the Laplacian of the smoothed image:



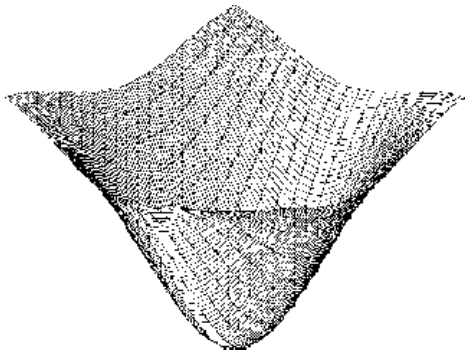
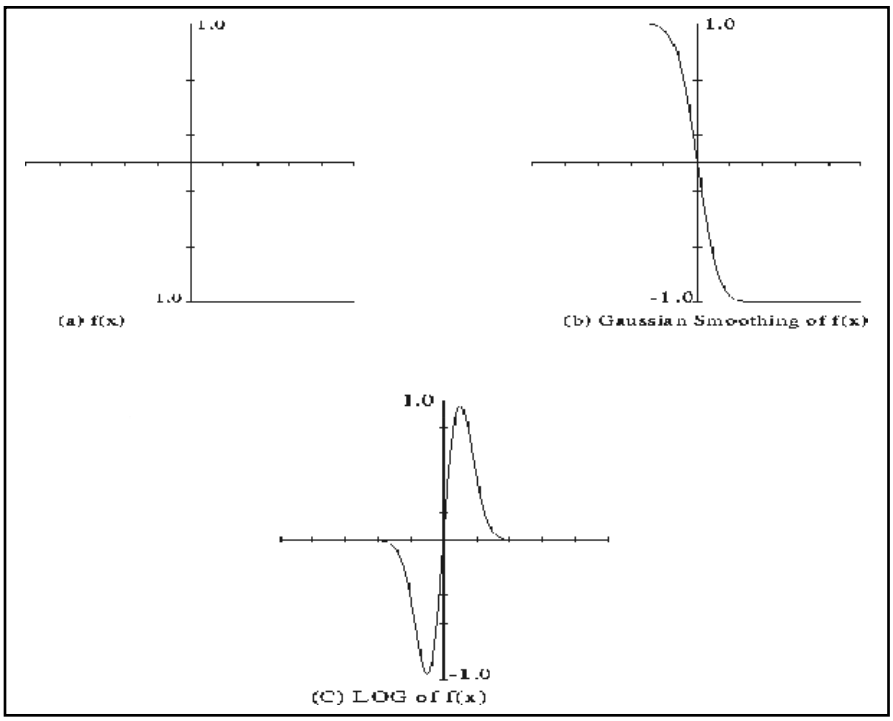


Fig 2.2 The Gaussian distribution in twovariable

Fig 2.3The LOG operator

Thus the edge pixels in an image are determined by a single convolution operation. This method of edge detection was first proposed by Marr and Hildreth at MIT who introduced the principle of the ***zero-crossing*** method. The basic principle of this method is to find the position in an image where the second derivatives become zero. These positions correspond to edge positions as shown in the fig 2.4



- The Gaussian function firstly smooths or blurs any step edges.
- The second derivative of the blurred image is taken; it has a zero-crossing at the edge. .

A related method of edge detection is that of applying the **Difference of Gaussian** (DOG) operator to an image.

- computed by applying two Gaussian operators with different values of to an image and forming the difference of the resulting two smoothed images.

- It can be shown that the DOG operator approximates the LOG operater.

Another important recent edge detection method is the **canny edge detector**. canny's approach is based on optimising the trade-off between two performance criteria:

- good edge detection -- there should be low probabilities of failing to mark real edge points and marking false edge points.
- good edge localisation -- the positions of edge points marked by the edge detector should be as close as possible to the real edge.

DIFFERENCE OPERATOR

an another simple operator that uses subtraction. the difference operator performs differentiation by calculating the differences between the pixles that surrounded the center of a

3 x 3 area. the difference operator finds the absolute value of the differences between opposite pixels, the upper left minus lower right, upper right minus lower left, left minus right, top minus bottom

3.2 THRESHOLD

These are based on the assumption that adjacent pixels whose value (gray-level, color value, texture, etc) lies within a certain range belong to the same class. thresholding techniques can obtain good segmentation of images that include only two opposite components. since these techniques neglect all the spatial relationship information of the images, they are inefficient for images that blur at object boundaries, or for multiple image component segmentation.

3.3 CONTOUR LINKING

Edge detectors yield pixels in an image lie on edges.The next step is to try to collect these pixels together into a set of edges. Thus, our aim is to replace many points on edges with a few edges themselves. The practical problem may be much more difficult than the idealised case.

- Small pieces of edges may be missing,
- Small edge segments may appear to be present due to noise where there is no real edge, ***etc***.

3.4 MATHEMATICAL MORPHOLOGY

We will deal here only with morphological operations for binary images. This will provide a basic understanding of the techniques. Morphological processing for gray scale images requires more sophisticated mathematical development. Morphological processing is described

almost entirely as operations on sets. In this discussion, a set is a collection of pixels in the context of an image. Our sets will be collections of points on an image grid G of size $N \times M$ pixels. A tool to extract description/representation of region shapes in an image.

In this, morphological operations can be used to correct the contour shape. Mainly there are four important techniques in mathematical morphology namely,

3.4.1 DILATION

It is used to add pixels to the boundary of objects and to fill in holes.

3.4.2 EROSION

It is used to reduce the size of an image by removing ‘on’ pixels from the boundaries of objects and also to increase the size of holes by removing pixels around the perimeter of the hole.

3.4.3 OPENING

More complex morphological operations are formed by cascading basic operations to obtain more complex operations. Application of the erosion operation followed by the dilation operation is called the **opening operation**.

3.4.4 CLOSING

The **closing operation** is defined as the dilation operation followed by the opening operation. A square or disk structure element results in smooth boundaries, fills small holes and joins narrow breaks.

3.5 CONTOUR LABELING

3.5.1 PIXEL CONNECTIVITY

Connectivity between pixels is an important concept used in establishing boundaries of objects and components of regions in an image. To establish whether two pixels are connected, it must be determines if they are adjacent in some sense and if their gray levels satisfy a specified criterion of similarity. For instance, in a binary image with values 0 and 1, two pixels may be 4 neighbors, but they are not said to be connected unless they have the same value. We consider two types of connectivity

1. **4-connectivity**

   Two pixels p and q with values from V are 4-connected if q is in the set N_4 (p)

$$N_4(p) = \{(x+1, y), (x-1, y), (x, y+1), (x, y-1)\}$$

2. 8-**connectivity**

   Two pixels p and q with values from V are 8-connected if q is in the set N_8 (p)

$$N_8(p) = N_4 \cup \{(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)\}$$

3.5.2 LABELING OF CONNECTED COMPONENTS

The ability to assign different labels to various disjoint, connected components of an image is of fundamental importance in automated image analysis. In the following section we develop a simple sequential connected components labeling procedure that operates on two rows

of a binary image at a time. Image scanning an image pixel by pixel, from left to right and from top to bottom and assume for the moment that we are interested in 4-connectivitycomponenets.

Let p denote the pixel at any step in the scanning process and let r and t denote the upper and left hand neighbors of p, respectively. The nature of the scanning sequence ensures that when we get to p, points r and t have already been encountered.

1. If the value of p is 0, simply move on to the next scanning position.
2. If the value of p is 1, examine r and t.
    a. If they are both 0, assign a new label to p.
    b. If only one of the two neighbors is 1, assign its label to p.
    c. If they are both 1 and have the same label, assign that label to p.
    d. If they are both 1 and have different same label, assign one of the labels to p and make a note that the two labels are equivalent

.

3. All we need to do now is sort all pairs of equivalent labels into equivalence classes, assign a different label to each to each class, and then do a second pass through the image, replacing each label by the label assigned to its equivalence class.

3.6 CONTOUR FILLING

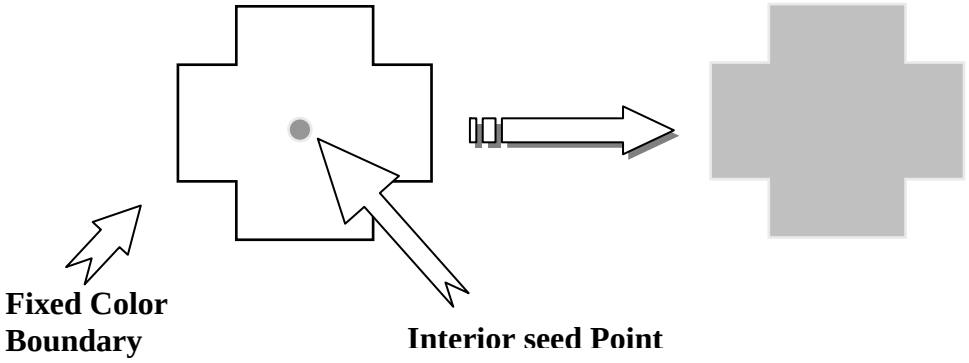
Sometimes, it is necessary to find the interior when a closed border is given and fill it with gray level of the border. From the standpoint of computer graphics, contour filling is

essentially a problem of vector to raster scan conversion. The contour is available either as a chain code of neighboring pels or as a polygon with given vertices.

3.6.1 FLOOD FILLING ALGORITHM#

In its simplest form, a flood-filling algorithm is one that changes the color of a region, given an initial pixel in that region. This is similar to the problem called “object labeling” in computer vision, which consists of assigning one same number to all the pixels in a region. The simplest algorithm that solves this problem is

Fig 2.6 Boundary Fill Algorithm



3.7 STUDYING BMP FILE FORMAT

The .bmp file format (sometimes also saved as .dib) is the standard for a Windows 3.0 or later DIB (device independent bitmap) file. BMP files are an historic (but still commonly used) file format for the historic (but still commonly used) operating system called "Windows". BMP images can range from black and white (1 byte per pixel) up to 24-bit color (16.7 million colors). While the images can be compressed.

SYSTEM STUDY AND REQUIREMENT SPECIFICATION

4.SYSTEM STUDY AND REQUIREMENT SPECIFICATION

4.1 PROBLEM FORMULATION

Intention is to extract and label the objects by numbers from the given multi spectral image.

4.2 SYSTEM STUDY

In this project, we are reading and storing image file, which is in the format of 24 bit or indexed image. First of all, we will input this image and read the image attributes. The next task is to convert these full color images into Edge pixel binary images, which only include black and white. We use improved edge detection to fulfill this request. We should select proper threshold so that the generated binary images have neither many loss of information nor many outliners. Here we are using the fast entropic threshold to generate the binary image. Then, we will divide the images into connected objects and label them by numbers.

4.2.1 IMPROVED COLOR EDGE EXTRACTION

Most existing isotropic edge detectors consider only the horizontal and vertical edge patterns. We develop an improved isotropic color-edge detector by including more potential edge patterns and integrating a fast entropic thresholding technique.

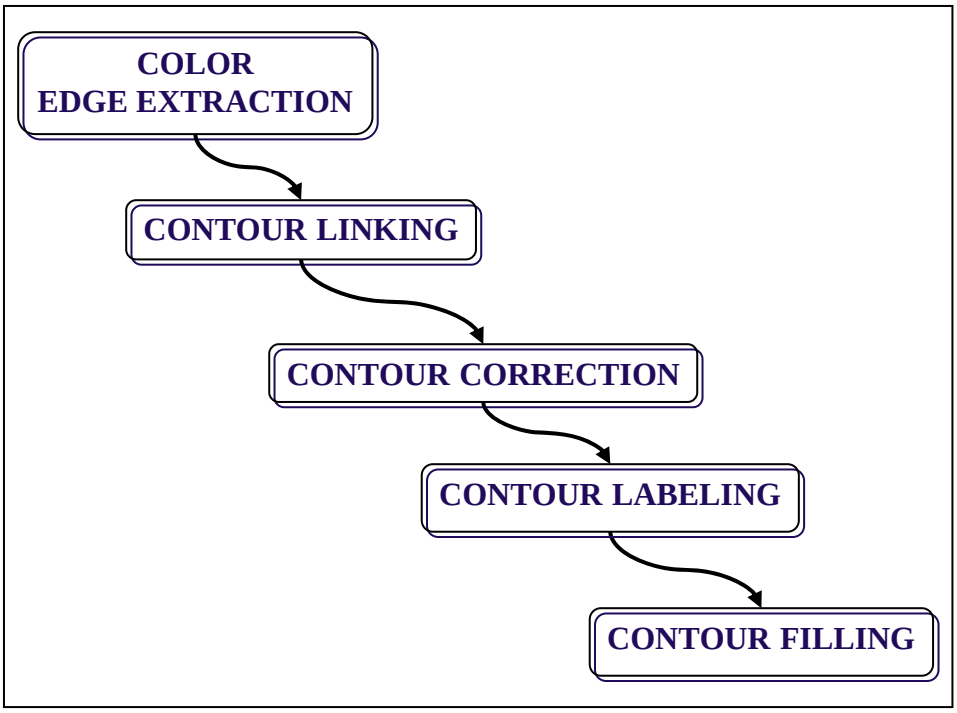


Figure 3.1

4.2.3 FAST ENTROPIC THRESHOLD

To automatically obtain an optimal threshold that is adaptive to the image contents, the *entropic thresholding technique* is used. To illustrate, let the local maximum edge strength of pixels have range [0, M].

In an input image, assume that there are f_i pixels whose local maximum edge strength has the value i, i € [0, M]. Given a threshold, e.g.,T , the probability distributions for the edge and nonedge pixel classes can be defined, respectively.

The probability for the nonedge pixels $P_n(i)$ can be defined as

$$P_n(i) = \frac{f_i}{\sum_{h=0}^{T} f_h}, \quad 0 \le i \le T$$

where $\sum_{h=0}^{T} f_h$ indicates the total number of pixels that have the local maximum edge

strength features in range 0< i < T.

The probability for the edge pixels can also be defined as

$$P_e(i) = \frac{f_i}{\sum_{h=T+1}^{M} f_h}, \quad T+1 \leq i \leq M \qquad \#$$

where $\sum_{h=T+1}^{M} f_h$ indicates the total number of pixels that have the local maximum edge strength

features in the range T+1 < i < M.

The entropies for these two pixel $\bar{T}$ classes are given below

$$H_n(T) = -\sum_{i=0}^{T} P_n(i) \log P_n(i)$$

$$H_e(T) = -\sum_{i=T+1}^{M} P_e(i) \log P_e(i). \qquad \#$$

The optimal threshold vector selected for performing the nonedge and edge pixel

classification has to satisfy the following criterion functions

$$H(\bar{T}) = \max_{T=0,1,2,\ldots,M} \{H_n(T) + H_e(T)\} \qquad \#$$

4.2.4 IMPROVED CONTOUR LINKING METHOD

In the above method, magnitude difference threshold and Angular thereshold is needed to link the particular pixel. But in this method, We have analysis the nonedge pixels with the edge pixels.

If the nonedge pixels found, then it check the 4 neightbors using 4-connectivity rule. If there is any edge pixles found, then makes the nonedge pixel to edge pixel. The algorithm terminates when all edge points have been linked to one edge or at least have been considered for linking once

4.2.5 MATHEMATICAL MORPHOLOGY

Morphological processing is described almost entirely as operations on sets. In this discussion, a set is a collection of pixels in the context of an image. Our sets will be collections of points on an image grid G of size $N \times M$ pixels. A tool to extract description/representation of region shapes in an image.

In this, morphological operations can be used to correct the contour shape

4.2.5.1 DILATION

It is used to add pixels to the boundary of objects and to fill in holes. The symbol and operation is shown below. In this, A is the image and B is the structuring element. The dilation of object A by object B is the union of all the possible translations of the object A by the elements in object B.

Dilation is defined as the union of the objects A1 and A2. NOT THE INTERSECTION.

$$A \oplus B = \{x : A \cap B_x \neq \phi\} \quad .$$

4.2.5.2 EROSION

It is used to reduce the size of an image by removing ‘on’ pixels from the boundaries of objects and also to increase the size of holes by removing pixels around the perimeter of the hole. The symbol and operation is shown below. In this, where A is the image and B is the structuring element. The effect of the morphological operation will depend on the shape of the structuring element. The erosion operation is the intersection of the translation of the object by a structuring element B, but the structuring element is first rotated 180 degrees about the origin before the object is translated.

$$A \oplus B = \{x : B_x \subset A\} \quad .$$

4.2.5.3 OPENING

More complex morphological operations are formed by cascading basic operations to obtain more complex operations. Application of the erosion operation followed by the dilation operation is called the **opening operation**. In the first phase, the erosion operation deletes pixels around the edges and noise pixels. In the second phase the dilation operation restores the objects to their original size.

$$A \bigcirc B = (A \ominus B) \oplus B$$

4.2.5.4 Closing

The **closing operation** is defined as the dilation operation followed by the opening operation. A square or disk structure element results in smooth boundaries, fills small holes and joins narrow breaks.

$$A \cdot B = (A \oplus B) \ominus B \quad .$$

4.2.6 CONTOUR LABELING

4.2.6.1 IMPROVED CONTOUR LABELING

In the conventional procedure, highly recursive algorithm uses great amount of memory, and cause stack overflow. So other algorithm based on coloring scan lines and using a stack has been developed. The following steps show the algorithm for improved border labeling.

PSEUDO PROCEDURE

1. Begin with an image array g comprising 0 and pixels set to a value.
2. Loops through the image array looking for a *g(i,j) == value*.
3. If finds such a pixel, then it calls the *checklabel* routine.
    i. Routine sets the pixel to *g_label* and examines the pixel's eight neighbors.
        1. If any neighbors ==l *value,* Push into the stack.
4. Control returns to the main algorithm,
    i. Each pixel on the stack is popped and sent to *checklabel* routine.
        1. All the points on the stack == *value,* so set them and check the neighbors.
5. After setting all the pixels in the first region, increment *g_label* and begin looking for the next region.

4.2.7 CONTOUR FILLING

4.2.7.1 IMPROVED CONTOUR FILLING ALGORITHM

In the flooding algorithm above, *propagation* started at a seed pixel in the region, and proceeded until reaching the region boundary.

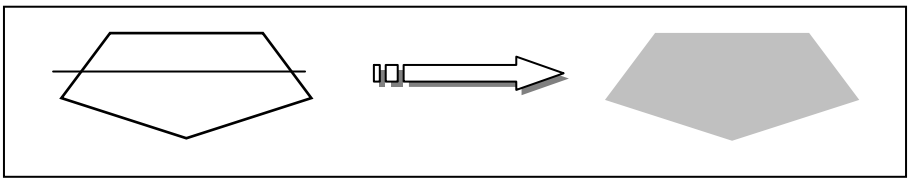
In the algorithms below, however, propagation starts at the boundary of the region and proceeds inwards.

To accomplish this, the boundary pixels of the region must be stored in a stack. For polygon representation, contour filling can be done by parity check algorithms. The basic idea of parity check is that a straight line intersects any closed curve an even number of times.

Major steps are as follows

1. Fill polygon interior row by row
2. Row-scan algorithm
3. Inside and Outside testing using parity checking algorithm

. This is shown in the figure fig 3.5



4.3 REQUIREMENT SPECIFICATION

According to the system study and problem formulation, the major requirement specifications for this project are;

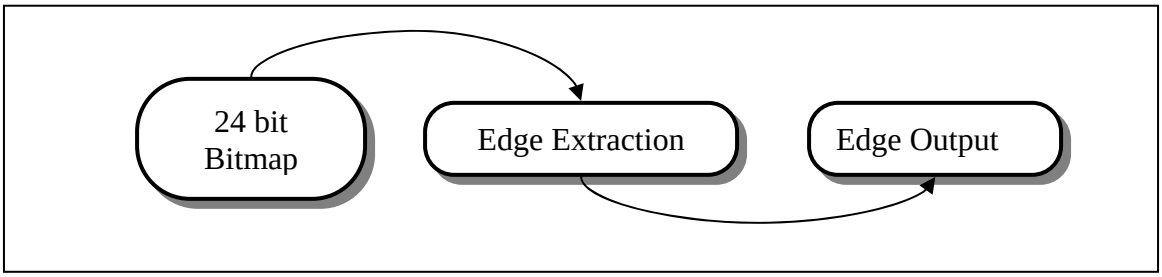
- ❖ Selection of an image
- ❖ Color edge extraction with linking
- ❖ Contour correction
- ❖ Contour labeling
- ❖ Contour filling

4.3.1 SELECTION OF AN IMAGE

In this, image analysis can be done in Window Bit map file format. There are many file formats for storing bitmaps, such as RLE, JPEG, TIFF, TGA, PCX, BMP, PNG, PCD and GIF. The bitmaps studied in this section will be 256-color bitmaps and 24 bit Bitmaps, where eight bits represents one pixel and 24 bit represents one pixel respectively. This file format can be stored uncompressed, so reading BMP files is fairly simple.

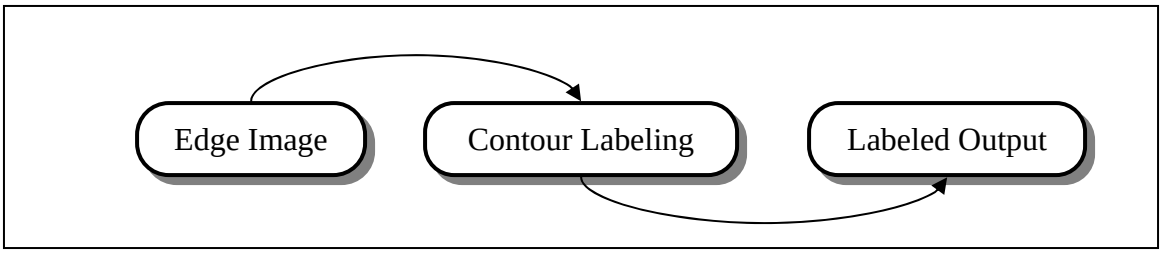
4.3.2 COLOR EDGE EXTRACTION WITH LINKING

In this, color edges can be extracted using improved isotropic edge detectors with fast entropic threshold. Therefore, the system has to convert input image to Edged image and this has been saved in a separate file and display it on the screen. The system process is depicted in the fig 3.6



4.3.3 CONTOUR LABELING

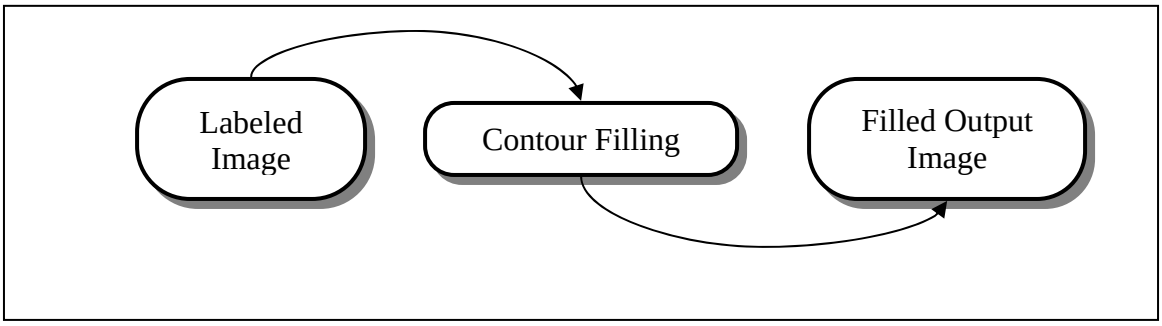
In this section, Edged output is going to be the input for the Labeling process. Therefore, the system has to convert input edge image to Labeled image and this has been saved in a separate file and display it on the screen. The system process is depicted in the fig 3.8



4.3.4 CONTOUR FILLING

In this section, Labeled output is going to be the input for the contour filling process. Therefore, the system has to convert input labeled image to filled image and this has been saved in a separate file and display it on the screen. This will be the final output of this project.

The system process is depicted in the fig 3.9



4.4.EXISTING SYSTEM

- ✓ Lot of research and progress have been made in the area of image segmentation.
- ✓ Robustness and generality of the algorithms on a large variety of image data have not been established well.
- ✓ major difficulties arises from the fact that most natural images are rich in color and texture.

- ✓ Content-based image retrieval (CBIR) systems have attracted the attention of both commercial developers and academic research community.
- ✓ Texture and color are important clues for CBIR systems, but the contribution of color to the texture perception has been mostly ignored
- ✓ Extracting individual image objects from an unconstrained image is difficult
- ✓ In gray scale images retrieval had been done
- ✓ Those methods were not applicable for color images.
- ✓ In color images only 16 and <16 bit colors are being extracted.

4.5.PROPOSED SYSTEM :

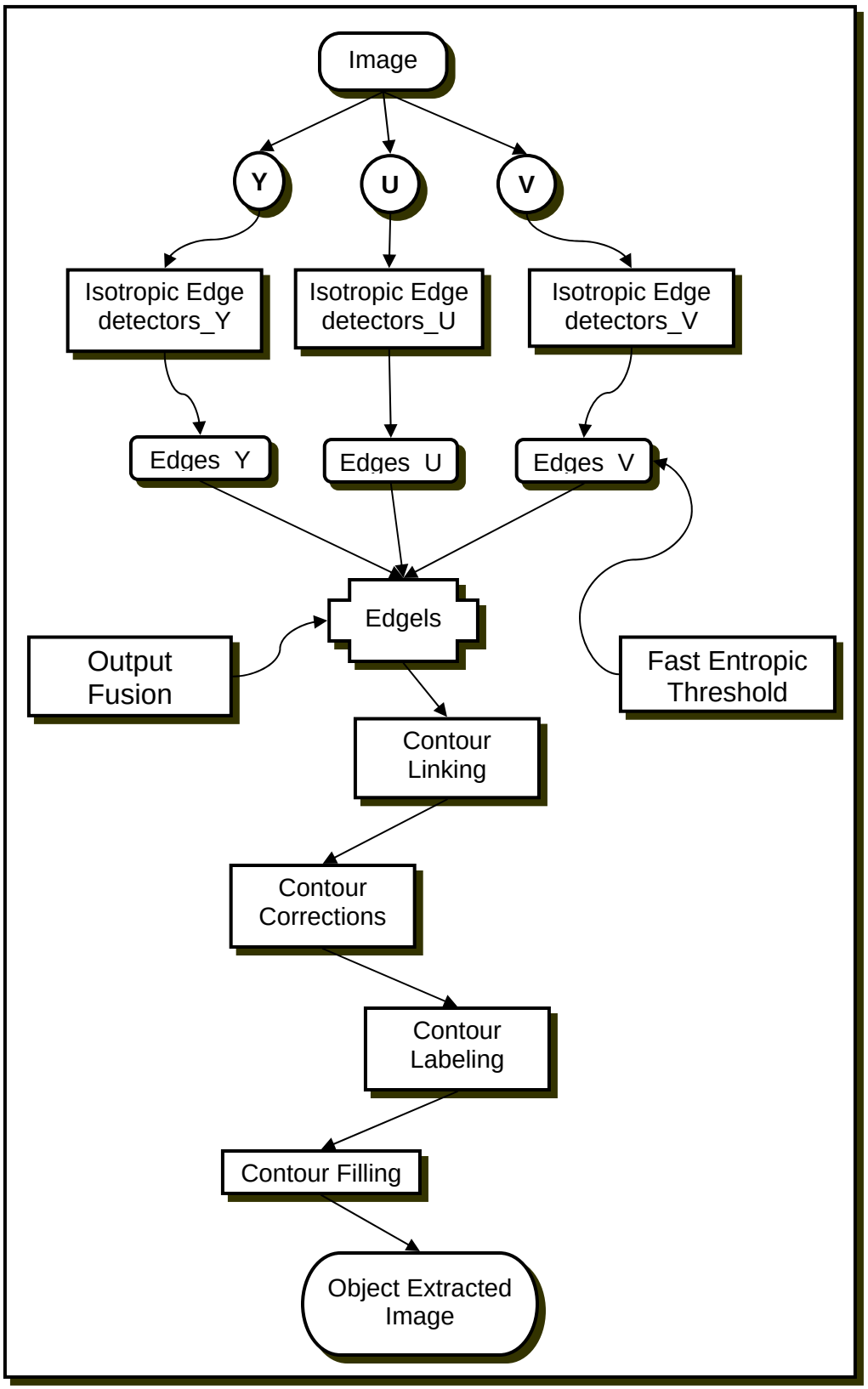
- ☐ A novel method of image extraction is proposed in this project.
- ☐ 24 and >24 bit color extraction is achieved in this project
- ☐ Extraction process is achieved by the following architecture

ARCHITECTURE DESIGN

5. ARCHITECTURE DESIGN

5.1 OVERALL SYSTEM DESIGN

The Complete system design architecture is shown in the fig 4.1. In this, Input image is going to be 24 Bit Bitmap image..



5.2 **I/O**

INTERFACE DESIGN

The I/O interface design consists of 10 parts as shown in fig 4.2. They are

- Selection for Edge detectors
- Selection for Thresholding
- Selection for Contour Linking
- Selection for Contour Corrections
- Selection for Contour Labeling
- Selection for Contour Filling
- Area for extracted objects
- Area for display input image
- Area for display Edge image
- Area for display Output image

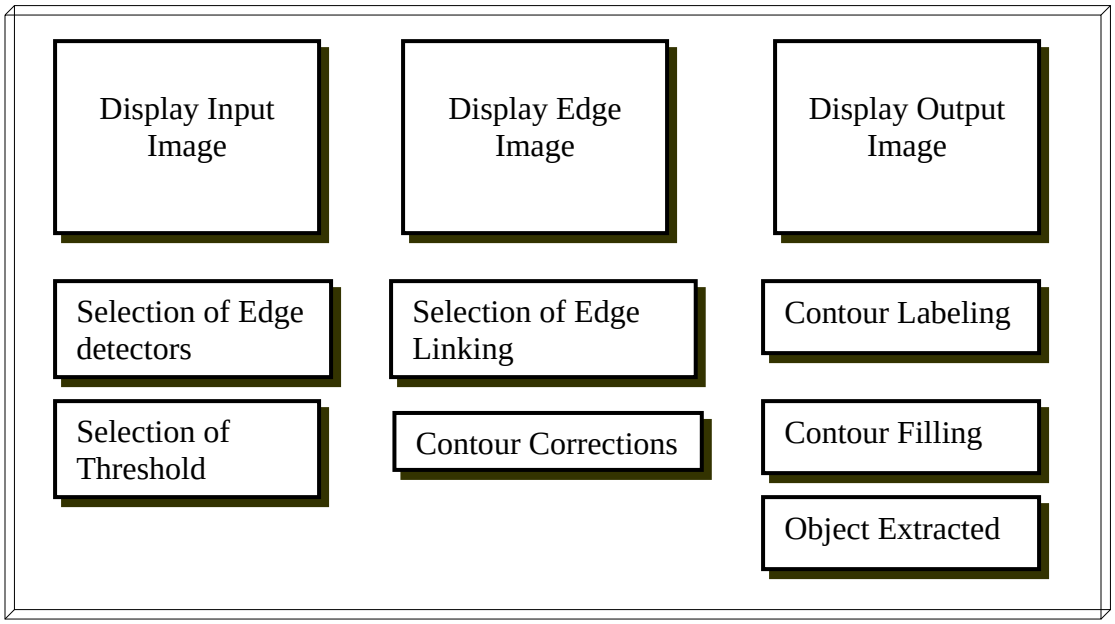


fig 5.2 I/O Interface Design

IMPLEMENTATION METHODOLOGY

6. IMPLEMENTATION METHODOLOGY

6.1 MODULES

The various modules involved in object extraction are

- Reading and displaying a 24 bit or 8 bit Bitmap image
- Color model Conversion to YUV from RGB
- Extracting Edges with Threshold
- Link the contour
- Contour correction
- Contour Labeling
- Contour Filling

6.1.1 READING AND DISPLAYING IMAGES

The input image given as input should be strictly 8-bit images or 24-bit color images. Any other images other than this will not be supported and leads to an error image.

Modules

- Function to Load the Image
- Function to Save the Image
- Function to convert the RGB color model to YUV color model
- Function to Display the image

6.1.2 EXTRACTING EDGES

The selected images are loaded and the information about the image can be stored in a single dimension array. It must me converted into two dimension array for easy manipulations. This has been done by Matrix ADT. Then, the resulting array can be given as an input for corresponding edge detection function. Traditional procedure as well as improved edge detector procedure has implemented for comparison purpose. The various types of edge detectors are shown below.

MODULES

- Function to convert Single Dimension raster data to Two Dimension raster data
- Function for Homogeneity Edge Detector
- Function for Improved Edge Detector

6.1.3 EXTRACTING EDGES WITH THRESHOLD

The Edged image is then thresholded with the corresponding threshold function. Traditional thresholding procedure as well as fast entropic threshold procedure has implemented for comparison purpose. The various types of threshold function are shown below

MODULES

- Function for finding threshold as conventional based
- Function for finding threshold as statistical based
- Function for finding a optimal threshold
- Function for finding a fast entropic threshold

6.2.4 CONTOUR CORRECTION

The Edged image can then fed into contour correction function. Mainly it can be applicable for index value bit map only and no need to apply for 24 bit Bitmap images.

MODULES

- Function for opening operation
- Function for closing operation

6.2.5 CONTOUR LABELING

The result of any successful image segment is the labeling of each pixel that lies within a specific distinct segment. The edged image can then fed into Contour labeling function.

MODULES

- Function for Simple labeling
- Function for Improved Contour labeling

6.2.6 CONTOUR FILLING

The labeled image can then fed into Contour filling function. The various types of function are listed

MODULES

- Function for Simple Distinct Objects Filling
- Function for Line Objects Filling
- Function for Complex Contour Filling

CODING AND TESTING

7. CODING AND TESTING

7.1 CODING RULE

It is essential that the overall strategy be completely mapped out before any of the detailed programming actually begins. This permits to concentrate on the general program logic, without being concerned with syntactic details of the actual instructions. The bottom up approach involves the detailed development of these program modules early in the planning process. The overall system development is then based upon the known characteristic of these available program modules.

The modules to satisfy the design specification and architecture design can be put into seven main categories, namely

- Loading, Saving and Displaying Bitmap images
- Various types of Edge detectors
- Various types Threshold technique
- Various types of Contour Linking
- Morphological Operations for contour corrections
- Various types of Contour Labeling
- Various types of Contour Filling

7.1.1 FUNCTIONS IN LOADING, SAVING AND DISPLAYING BITMAP IMAGES:

- BIT LImage(char *)
  - Function is used to load the bitmap file header, bitmap information header, Color table, and raster data.
  - Input signature is the filename to be loaded

- BIT SaveImage(char*)
  - Function is used to save the image in the disk.
  - This function is called internally from the LImage function.
  - Input signature is the filename to be loaded

7.1.2 FUNCTIONS IN EDGE DETECTORS

- void GraEdgedet()
  - Function is used to extract the edges which is based on the gradient into two directions
  - Internally it reads the image raster and apply the convolution procedure to it

- void HomEdgedet()
  - Function is used to extract the edges which is based on the Homogeneity operation
  - Internally it reads the image raster and apply the convolution procedure to it

- void DifEdgedet()
  - Function is used to extract the edges which is based on the Difference operation
  - Internally it reads the image raster and apply the convolution procedure to it

7.2 TESTING

Programming errors often remain undetected until an attempt is made to compile or execute the program. The presence of syntactic (or grammatical) errors will become readily apparent once the RUN command has been issued, since these errors will prevent the program from being compiled or executed successfully. Some particularly common errors of this type are improperly declared variables, a reference to an undeclared variable, incorrect punctuation, etc.

Most compilers will generate diagnostic messages when syntactic errors have been detected during the compilation process. These diagnostic messages are not always straightforward in their meaning and they may not correctly identify the nature and the approximate location of the errors. If the program includes several different syntactic errors, they may not all be detected on the first pass through the compiler. Thus, it may be necessary to correct syntactic errors before other can be found. This process could repeat itself through several cycles before all of the syntactic errors before have been identifies and corrected.

CONCULSION AND FUTURE IMPROVEMENT

CONCLUSION AND FUTURE IMPROVEMENT

8.1 CONCLUSION

The proposed automatic object extraction in a multi spectral image provides a basis for applications such as object based image searching, browsing and retrieving as well as object based image indexing for effective management of large image collections. The improved isotropic edge detector with fast entropic threshold based approach has several advantages over the conventional methods. They are:

- Over-segmentation or under-segmentation either increases the computation complexity of image labeling or results in inaccurate labeling results when small objects in images are hidden by the under segmentation.
- Simple thresholding results in objects with small region size “eaten” by neighboring objects with larger regions size.

8.2 FUTURE WORK

Potential extensions of this work may be:

Extend to digital video object labeling for video content analysis, object tracking and motion analysis

BIBLIOGRAPHY

BIBLIOGRAPHY

- Wei-Ying Ma and B.S.Manjunath, “Edge Flow: A technique for Boundary Detection and Image Segmentation”, IEEE Trans on Image Processing, Vol 9, No.8 August 2000

- Jaime Silvela and Javier Portillo “Breadth-First Search and its Application to Image Processing Problems”, IEEE Trans on Image Processing, Vol 10, No.8 August 2001

- Nawapak Eua-Anant, and Lalita Udpa “Boundary Detection Using Simulation of Particle Motion in a Vector Image Field”, IEEE Trans on Image Processing, Vol 8, No, 11 November 1999

- Jianping Fan, David.K.Y. Yau, Ahmed.K.Elmagarmid, and Walid G.Aref “Automatic Image Segmentation by Integrating Color Edge Extraction and Seeded Region Growing” IEEE Trans on Image Processing, Vol 10, No.10 October 2001

- Chao Han, Thomas S.Hatsukami, Jenq-Neng Hwang, and Chun Yuan “A Fast Minimal Path Active Contour Model” IEEE Trans on Image Processing, Vol 10, No.6 June 2001

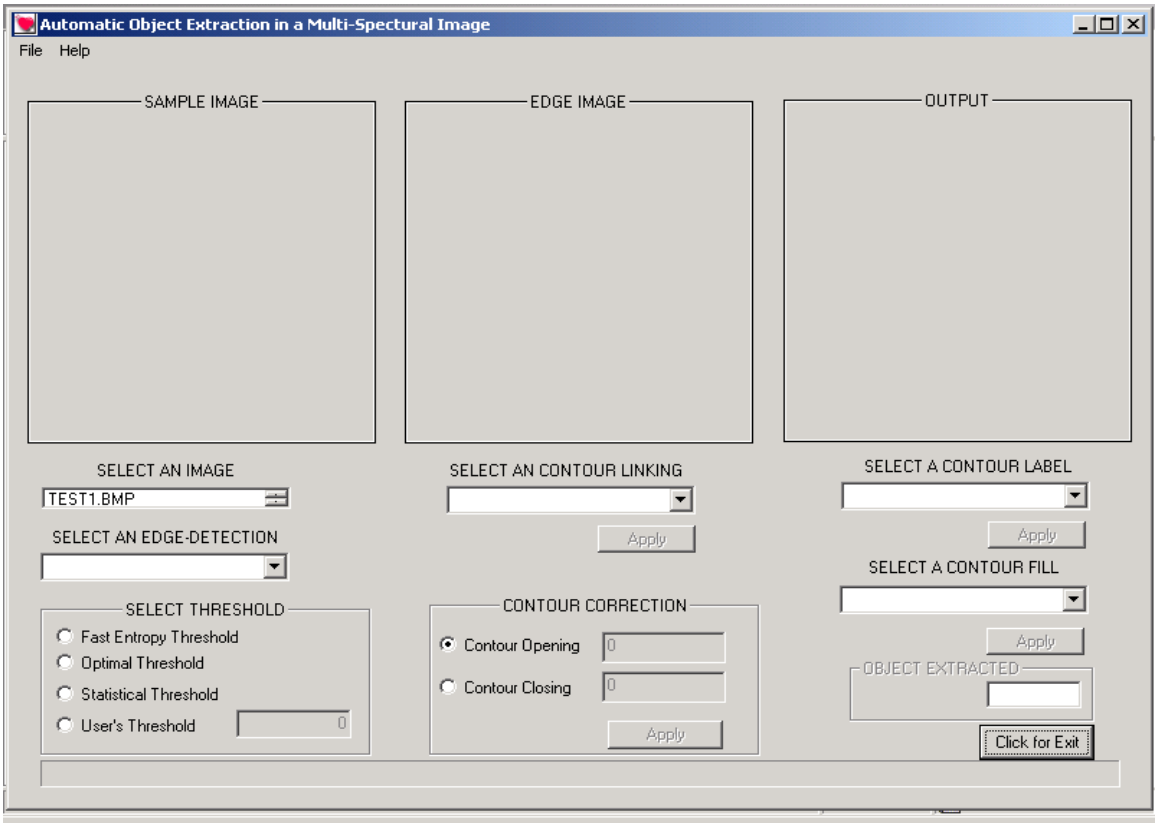
📖 Rafael C.Gonzalez, Richard E.Woods “Digital image Processing”

APPENDIX

SCREEN SHOTS

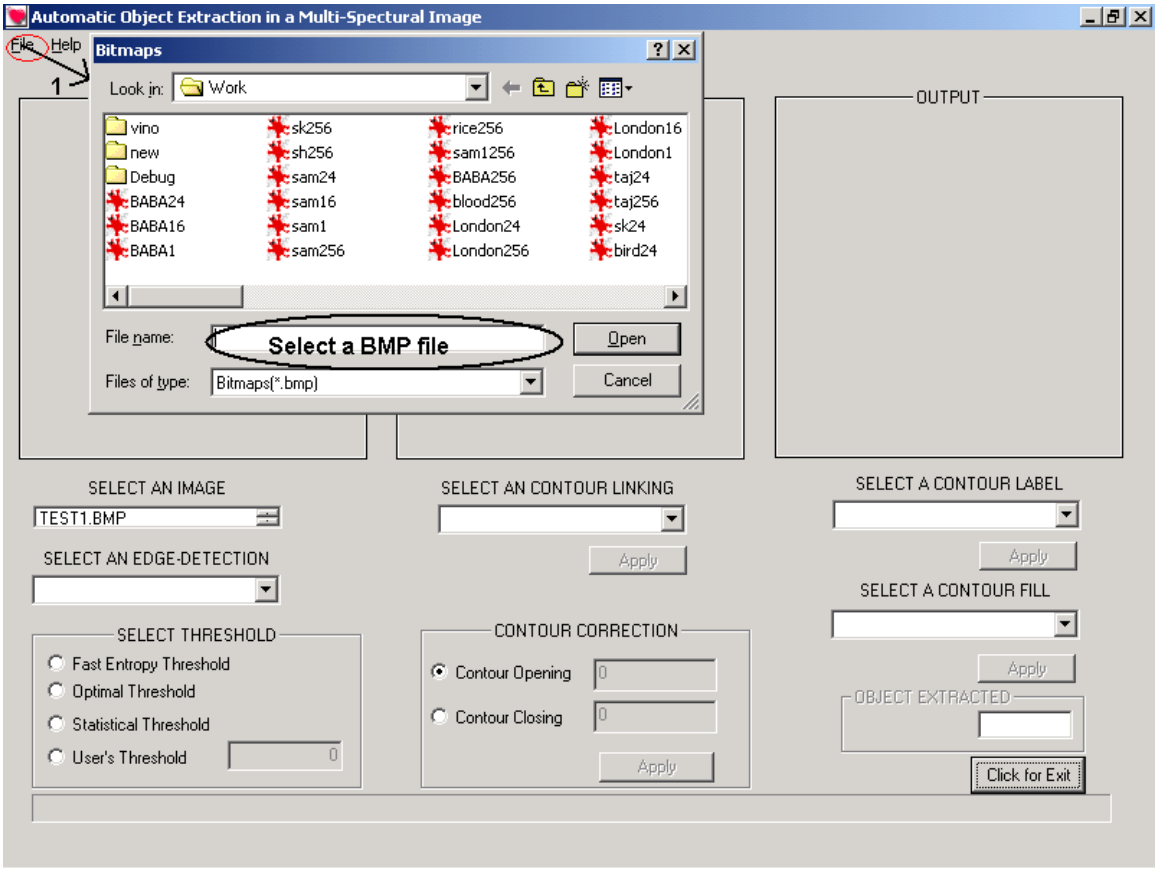
USER'S MANUAL

The front end design is designed such that it must be user friendly. The front end design of the system is shown in *fig 7.1*.



The following steps to be followed for using the software and this is shown in the figure which is self explanatory.

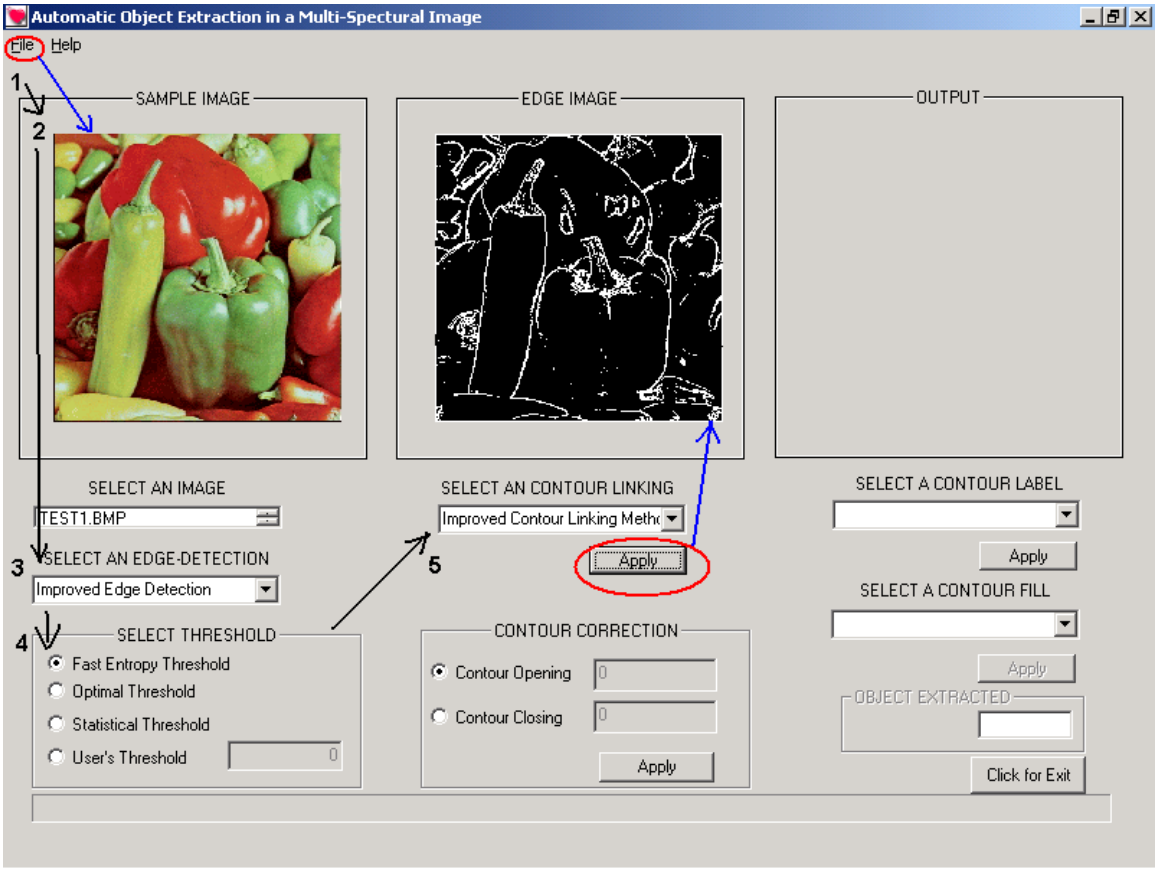
1. First, select the image from the ***file*** menu



2. Input image is displayed on the screen
3. Select the ***edge detectors*** from the list box
4. Select the ***Threshold*** type
5. Select the ***Contour linking*** from the list box

Press ***Apply***

The Resulted output is shown in the Edge Image screen

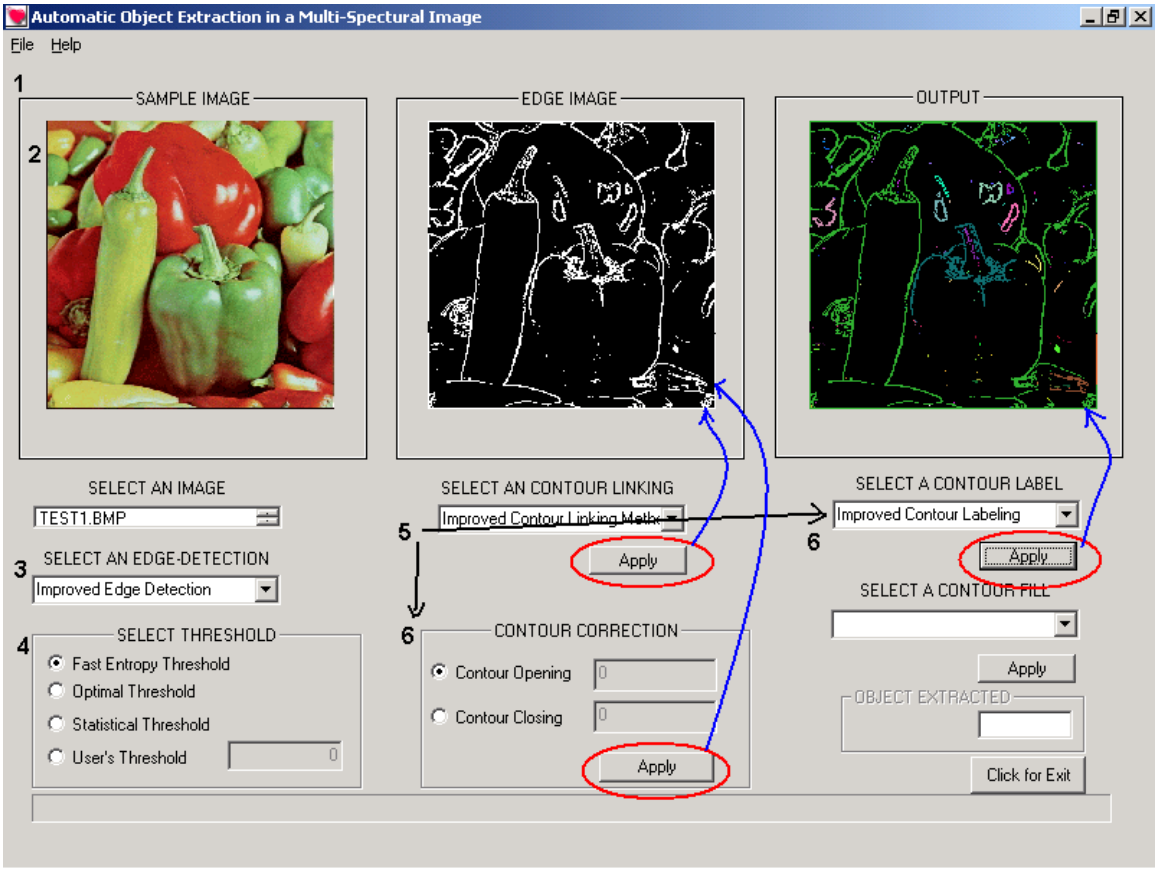


6.Select the ***contour correction*** and give the value in the text box and it is applicable only for index table image.

7.Select the ***contour label*** from the list box

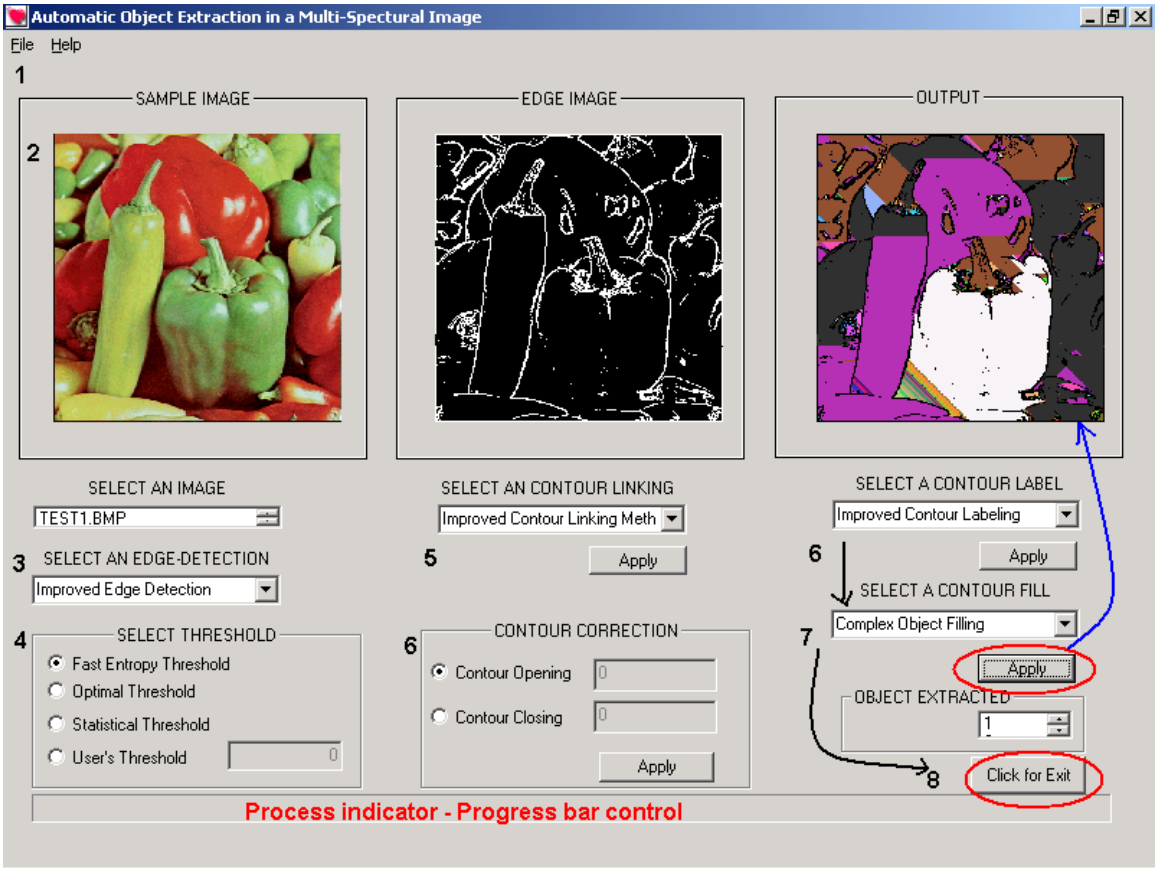
Press ***Apply***

The Resulted output is shown in the Output Image screen



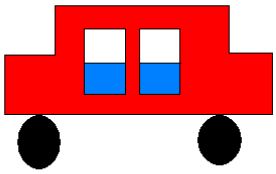
8.. Select the ***contour filling*** from the list box
Press ***Apply***

The Resulted output is shown in the Output Image screen

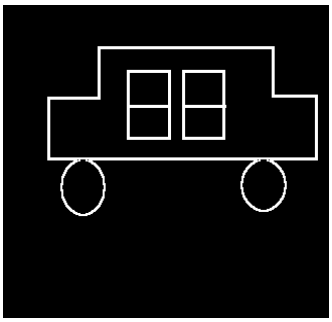


7.1 SAMPLE OUTPUT

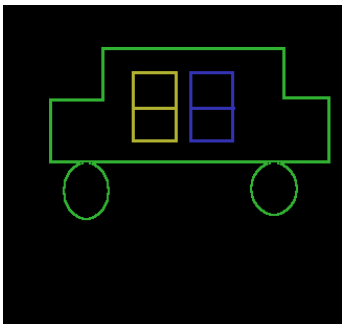
Input Image



Edge Image



Contour Labeled Image



Contour Filled Image