A KNOWLEDGE-BASED MACHINE VISION SYSTEM

Ning Lu

A thesis submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg, in fulfillment of the requirements for the degree of Doctor of Philosophy.

Johannesburg, 1997

DECLARATION

I declare that this thesis is my own, unaided work. It is being submitted for the degree of Doctor of Philosophy in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

5th day of Dec 1997

ABSTRACT

As a result of the work on this project, two machine vision systems are presented in this thesis. The first system is designed for an industrial application which identifies and locates machined components which have general geometric boundaries. The system consists of a CCD camera, a Pi030 image processing system, and a host computer running under the Linux operating system. The problem of extracting the shape information is treated as a two level image processing procedure, viz. Segmentation and decomposition. Objects are first segmented into generic 'blobs' and each blob is further decomposed into a primitive chain which constitutes a series of dominant points and basic primitive segments. The recognition task is accomplished by applying the knowledge base techniques. The shape information is used to form an object description using a specific scheme which is a combination of a semantic network, feature vectors and a production system. The knowledge base is constructed with this description scheme. The inference engine is based on an object verification procedure and a blob verification procedure. The inference results are also used as feedback information to improve the image processing procedure. Experimental results have proved that this system has the advantages of accuracy and reliability. In tests conducted on 19 machined components with different combinations of generic features, the classification system showed a 100% success rate. In addition the system is capable of being easily modified and expanded to cater for parts that are touching and/or overlapping as well as parts in motion on a conveyor.

The second machine vision system has been separately designed to carry out the fruit inspection and classification tasks automatically. The system uses a low cost Personal Computer (PC) and an image acquisition hardware to perform these tasks. The concept of fuzzy sets is employed to design the fruit classifier. Preliminary results have proved the feasibility of the system which is felt to be suitable for individual farmers who wish to carry out automatic fruit classification.

In Memory of My Father

Lu Luren

1922-8-5 to 1994-2-28

ACKNOWLEDGMENTS

What I cherish most in my life is the strange luck God bestowed on me: there are always nice people around me. This strange luck made my stay in South Africa more of a happy time, than a lonely, isolated struggle.

The freedom, the trust, the support and guidance I enjoyed while I was doing my thesis research combined to be a force pushing me ahead through the hard time of thinking and programming. I owe them to my supervisor, Mr. E. R. Fielding.

Through his intellectual insights, suggestions and comments, I have learned to see things from different angles. I would like to express my sincere gratitude to my co-supervisor, Dr. Vladimir Nedeljkovic.

I also want to thank Mr. Alan Tredgold, who always generously gave his precious time to help me on conducting experiments, discussing my work, editing my writings whenever I was entangled with difficulties. I enjoyed working with him. The days are gone, but not the memory of chatting, the hiking, the Good Hope Cape touring.

My sincere appreciation goes to Mr. H. D. Smith for his excellent job of manufacturing mechanical parts for my experiments.

Thanks are also given to FRD and Wits University for financial assistance.

Last, but not least, I would like to give my thanks to my family, for their continuous support and encouragement.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 An Overview of Machine Vision

Throughout the history of civilization there has been constant effort to design machines capable of doing work for mankind. Modern machines and programmable manipulators (industrial robots) have been designed and relieved humans from heavy, dangerous work and greatly increased production efficiency. However, even industrial robots have been restricted to performing important, but fairly simple manipulative tasks, such as lo.. ng and unloading other machines, tacking parts, spot-welding, paint-spraying and so forth. Any machine would be more intelligent and have greater powers of versatility if it could acquire information about its environment through its own vision system rather than being limited to a knowledge base provided by its programmer. The active research area of machine vision is the direct result of the desire of making a machine capable of seeing and interacting with the real world.

Machine Vision, also called Computer Vision, is referred to as automatic computer analysis of images for recognition, inspection and verification. The science of machine vision can be regarded as two fundamental tasks: one is gathering all the information needed to understand an image, the other is object recognition and classification.

A typical machine vision system consists of an image device and a computer. The image device scans a scene, its output is converted to a digital code by an A/D

converter and stored in memory as a digital image. The computer then analyzes the digital images and applies various techniques to enable the scene to be interpreted.

The first task of a machine vision system basically deals with image processing, which is performed to enhance the quality of images or to obtain a more suitable image representation. It generally consists of the following steps:

- Image Acquisition
- Feature Enhancement
- Image segmentation
- Image Interpretation
- Display, Compression and Storage

Interest in digital image processing techniques dates back to the early 1920's, when digitized pictures of world events were first transmitted between New York and London via the Bartlane cable picture transmission system. From then on great advances have been developed in hardware, processing techniques and methodologies.

Object recognition and classification is basically a labeling process that assigns an identity to each object. It is closely related to the field of pattern recognition which has been highly developed in theoretical aspects over the succeeding decades. There are two dominant models: the feature extraction-classification model and the linguistic or syntactic model.

1 INTRODUCTION

The Fuzzy Set model proposed by Zadeh (Bellman, Kalaba and Zadeh,1966) was first applied to classification in a theoretical paper (Zadeh, 1976) in 1966. Fuzzy Sets allow a gradual rather than an abrupt transition from membership to membership. Its intrinsic use of vagueness of an observed pattern and its evaluation procedure based on a fuzzy set theoretic method and inference, which only consists of minimum and maximum operators, shows promise in solving industrial engineering problems.

Artificial Intelligence is used to analyze scenes by computing a symbolic representation of a scene's contents after the images have been processed to obtain features. Artificial Intelligence may be viewed as having three stages: perception, cognition, and action. Perception translates signals from the world into symbols, cognition manipulates symbols, and action translates symbols into signals that effect changes in the world. Therefore, machine vision is also considered as a major discipline derived from artificial intelligence research.

To know the "essence" of an object or pattern, the science of psychophysics, along with cognitive science, which studies human vision, has been developed. Many techniques in machine vision are related to what is known about human vision.

Machine vision has the ability to revolutionize industrial engineering by its application in inspection operations such as printed circuit board inspection and metal finish inspection, and in adaptive control for robots in operations such as packaging, seam tracking in arc welding and assembly. Recently some machine

vision systems have been developed for the food processing industry inspired by the advantages of non-contact inspection.

However, for a machine to truly see a scene and use the knowledge obtained from the scene for future problem-solving tasks, has been proved to be, and will still remain, as one of the most difficult task for computers. Most machine vision systems are very application oriented, and many of them are still at the laboratory research stage. The reasons are mainly due to the complexity of any industrial scene, the processing time limitation of hardware and the cost.

Therefore, the question that is posed is how to use the sheer power of computers to solve such a complicated problem that has so far seemed beyond our reach? We may get some enlightenment from the following story.

1.2 Enlightenment from Deep Blue

On May 11 1997, the world witnessed the introduction of a new and powerful tool at the chess table. In a shocking finale that lasted barely more than an hour, World Champion Garry Kasparov resigned 19 moves into Game 6, handing a historic victory to Deep Blue

The win gave the IBM supercomputer a 3.5 to 2.5 victory in the six-game rematch. It was the first time a current world champion has lost a match to a computer opponent under tournament conditions.

While the match between Garry Kasparov and Deep Blue was no doubt an interesting and exciting match to watch, the game of chess itself is only a small

part of a much larger picture. At the heart of the event is an important computer science experiment being conducted by the Deep Blue development team.

Firstly, we have to realize that we should try to exploit the particular strengths of machines to solve our complex problem. Computers are extremely fast and have superb memories. Deep Blue has special-purpose hardware that enables it to calculate nearly a quarter of a billion chess positions per second.

In chess games, it is an axiom that if more moves ahead can be anticipated by a player than by his opponent, then that player will be the stronger. Deep Blue accomplishes this goal by a rapid and massive parallel search -- in other words by brute force methods.

As early as the 1950s and 1960s, researchers sought to design the chess systems by duplicating or mimicking the methods of a human player. Many researchers at that time maintained that computers should be programmed in sophisticated programming languages that would make it easy for programmers to incorporate the thought processes of grandmasters into their programs. It was expected that the computer would be able to beat a human master in a short space of time, however 40 years has passed since that initial speculation. Deep Blue has followed a different path: it has been programmed in C and taken the brute force approach.

Secondly, the importance of artificial intelligence has been appreciated. Even though a computer is empowered by using rapid and massive parallel search techniques, it still has the weakness of being poor at pattern recognition, complex strategy and adaptability.

Garry Kasparov stated quite clearly that his tournament victory over Deep Blue in February 1996 came because he could analyze the first game and play to exploit Deep Blue's evident weaknesses in the next. Conversely, Kasparov changed his own style of play in the middle of games, and Deep Blue was not prepared or able to adapt accordingly.

In this year's rematch, only the software of Deep Blue has been improved. In some aspects, the Deep Blue system has employed crude versions of methods we know are important in human intelligence. That is why Kasparov doubted that there was human interfacing during this rematch. He could not believe computer could play in such a way.

Lastly, the lesson of the importance of knowledge was learned. The victory of Deep Blue was heavily dependent upon the large amount of the recorded chess games accumulated during the last forty years. In some cases, even the fastest computer can become entangled with the "exponential explosion " problem, when a further deep move search proceeds. In this case the recorded games assisted Deep Blue to sift out some unnecessary searches.

This is the most impressive enlightenment learned from Deep Blue. It forced us to consider the important role of knowledge and knowledge representation techniques during design of this machine vision system.

1.3 Aims of the Study

The emphasis in machine vision systems is on maximizing the automatic operation at each stage. This means that real time operations and no human

interfacing are required. To accomplish these requirements, knowledge must be provided for the machine to make its own decisions. Without explicit use of knowledge, machine vision systems can be designed to work only in a very constrained environment for very limited applications. The more knowledge provided, the more intelligent a machine.

From studying the human learning process, we know that knowledge is accumulated through experience and learning. Deep Blue could not defeat the human grandmaster without the advantage of nearly 40 years of accumulation of large amounts of recorded games.

It is this motivation that is used to solve the recognition task performed by the machine vision system utilizing knowledge base techniques.

Another problem found in most machine vision systems, is that they usually assume the two basic tasks of machine vision -- gathering information from the image and object recognition and classification, are separate independent stages. This may explain why there are no satisfactory image processing techniques which are generally successful even after decades of research effort. In general, the object contours cannot be completely or accurately extracted from the image.

The assumption made here is that the whole image processing and recognition processes should be integrated and that the information from the later recognition stage can serve as a kind a knowledge and feedback to aid the accomplishment of the information extraction task.

Based on the above considerations, a knowledge based machine vision system, has been designed and implemented which is capable of recognizing and locating mechanical parts in a digitized image. The goals of this study can be summarized as follows:

- Extract shape information by a two-level image processing procedure,
- Design a shape description scheme,
- Build a knowledge based system to accomplish the recognition task.

1.4 The perspective and Organization of the Thesis

Following this chapter a system overview is presented in chapter 2, and the discussion on image acquisition in chapter 3. A detailed explanation of how to implement the user interfacing function with the system is also given.

The proposed system accomplishes the task of image information gathering by performing two levels of image segmentation operations. Chapter 4 deals with the high level image segmentation task. It starts with a brief review of the image segmentation techniques and then gives a full description on how to extract blobs and their edges from an image. Computations on geometric features and relationship features are also discussed.

After the high level image segmentation, the blobs which have been singled out, will undergo a further decomposition operation. This is the topic of chapter 5. After giving a full discussion on the advantages and disadvantages of current primitive extraction approaches, a simple decomposition scheme for the system is

proposed. Details are also given on the implementation of this decomposition scheme. Methods of how to use the feedback information to help the merging operations are also explained.

After the two-level image segmentation operations, the system proceeds to the object recognition stage. Chapter 6 describes how to use a knowledge based system to accomplish this task. It starts with an investigation of shape description methods and explains how to use the extracted information from the two-level segmentation to generate an appropriate description on objects and blobs in the system. The shape descriptions will be used to construct the knowledge base. Following a survey on pattern recognition techniques, an approach based on object verification and blob verification is proposed for building the inference engine. Reasoning strategies behind the inference engine are also explained.

Our conclusion in chapter 7 is based on a comparison with some typical systems which were in the same application domain. After a discussion on the achievements and weakness with this system, it suggests future work which should be done on this system.

Chapter 8 describes a separate machine vision system specially designed to solve the fruit inspection and packaging task. It uses the fuzzy set theory to assist soft fruit classification. Details are given on image acquisition, image segmentation, design of the fuzzy classifier and a learning algorithm. Experimental results are also presented and discussed.

2. SYSTEM OVERVIEW

This chapter commences with a description of the equipment configuration, together with a brief description of each element of the system. An explanation of how the system memory is managed is given, and finally the conventions used in the thesis are explained.

2.1 Equipment Setup

The basic elements of this system are a camera, the Pi030 Vision System, a Colour Monitor, mouse, and a host computer workstation. (the arrangement of these elements is shown in Figure 2-1)

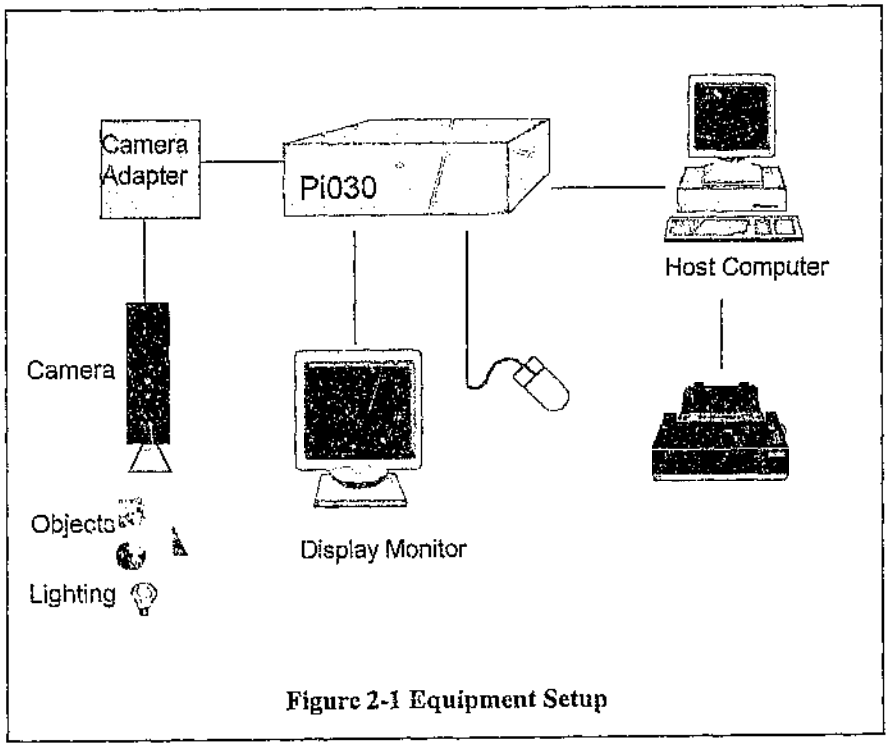


Figure 2-1 Equipment Setup

2.2 Camera

The camera in use is a Sony XC-77CE black and white CCD video camera. It is connected with a Sony DC-777 camera adapter. The video signal from the camera adapter is sent to the Pi030 machine through the camera port. An external sync signal can be connected with the camera adapter if external sync is required.

2.3 Monitor and Mouse

A colour monitor is used to observe the image processing process. Figure 2-2 illustrates the image flow from the Pi030 vision system to the monitor.

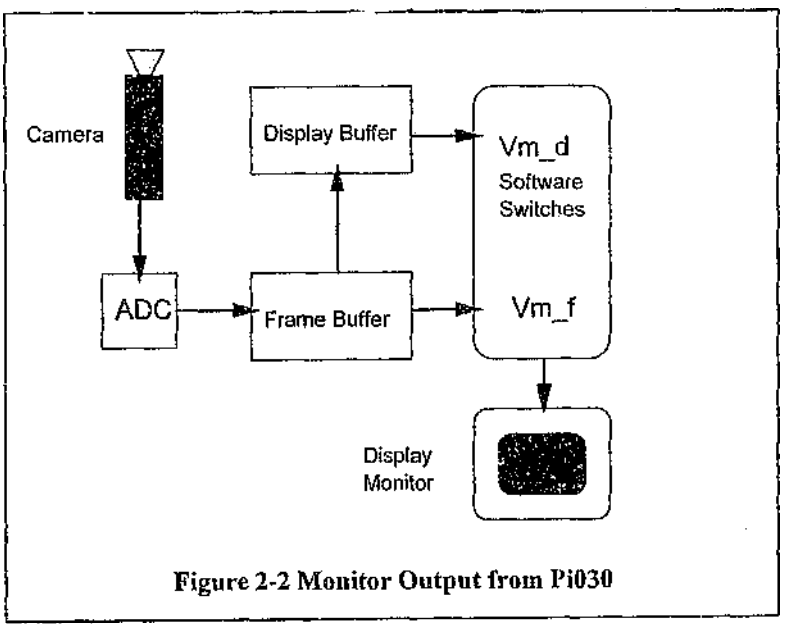


Figure 2-2 Monitor Output from Pi030

An NEC Multisync 3V monitor was specified instead of a VGA monitor. The reason for this choice being that a Multisync monitor allows a live image to be

displayed directly from the Frame Buffer. The VGA monitor is only capable of displaying an image from the display buffer and therefore the live image can only be displayed on a VGA monitor by writing a loop software routine which snaps an image and then copies it to the Display Buffer. Even though a VGA monitor is cheaper than a Multisync monitor, the provision of this additional routine is inconvenient in developing a vision system and not worth the saving in monitor cost.

In order to interface with the image processing process, drop-down menus are used and displayed on the color monitor. To manipulate the menus, a mouse is required. which can be connected to any RS232 port except the first port, which is reserved for the Console TTY.

2.4 Pi030 Vision System

The Pi030 Vision System is supplied by Performance Vision Ltd. This hardware has been incorporated into our system because a range of standard image analysis functions are provided as embedded functions which run at high speed. This feature makes real time industrial applications possible.

2.4.1 Hardware

The high image processing performance of the Pi030 Vision Machine is primarily obtained by the use of a dedicated Image Analysis Processor (IAP). The IAP is a pipelined array processor, the operations of which are controlled by microprograms.

2.4.2 Memory

The Pi030 is supplied with 1 Megabyte of main memory (DRAM) and 4 megabytes of frame buffer (see Figure 2-3).

The hardware allows the content of every window in the frame buffer to be displayed directly on the monitor. Since the images are displayed directly, the image buffer consumes part of the "bandwidth" of the image buffer memory, which is therefore not available for the processing. A special hardware Display Buffer is provided which is solely dedicated to the display of images. This feature makes it possible to have certain images constantly displayed on the monitor, which enables operators to maintain a continuous surveillance, thus ensuring the system is functioning correctly during the production mode.

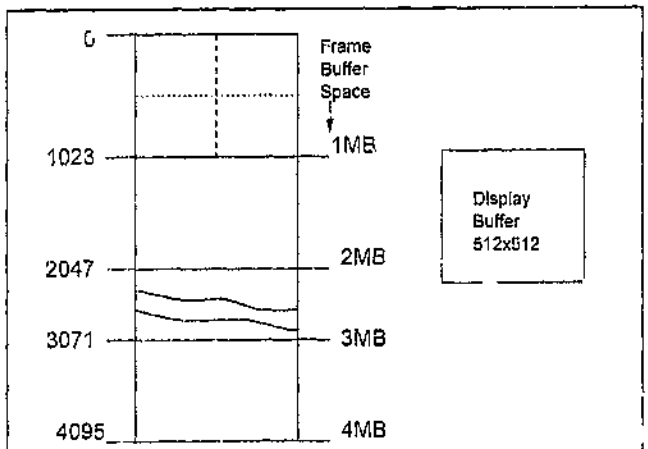


Figure 2-3 Pi030 Frame Buffer

2.4.3 System Software

2.4.3.1 The PICNIX Operating System

The Pi030 Vision System is provided with its own real-time operating system, PICNIX. This system services general industrial application requirements such as real time image analysis processing and networking with other computers to perform multiple tasks.

The PICNIX operating system is programmed on an EPROM which is located on the main board and installed by the manufacturer.

2.4.3.2 Software Development Environment

The Pi030 is also provided with a TOOLCHAIN software development environment for the host computer. **Toolchain** is a complete program **cross development** system with all the software utilities for developing and maintaining application programs and the system's libraries.

A **cross development** system is used for the circumstances, when applications are developed on one family of microprocessors, for a system running on another family of microprocessors.

2.5 Host Computer

The host computer is connected to the Pi030 Vision System via an Ethernet. The reasons for connecting the Pi030 to a microcomputer workstation is to take advantage of the comfortable and efficient program development environment provided by modern workstations. In addition it is very much more convenient to

explore the Image Analysis Library supplied by Pi030 and develop new application programs.

The host computer is an 80486 microcomputer running the Linux operating system as the program development environment. Linux was chosen for program development as it is a freely-distributable implementation of UNIX for 80386 and 80486 machines and it supports a wide range of software. The Pi030 TOOLCHAIN program development software is downloaded onto the host computer.

The entire application program is written in the C language. The reasons for using C as the programming language are:

- The Pi030 system software is based on the use of C language.
- The examples provided by the Pi030 are in C source files.
- C is a programming language which has close similarities to the low level programming language, Assembly, which can easily access the system hardware.

2.6 Memory Management

The Pi030 is provided with 4 Megabytes of frame buffer memory, however, most operations are carried out using only the first Megabyte of this allocation. The first Megabyte of the frame buffer is divided into four equal parts, and assigned as **the display, work 1, work 2, and camera** segments as shown in Figure 2-4. In

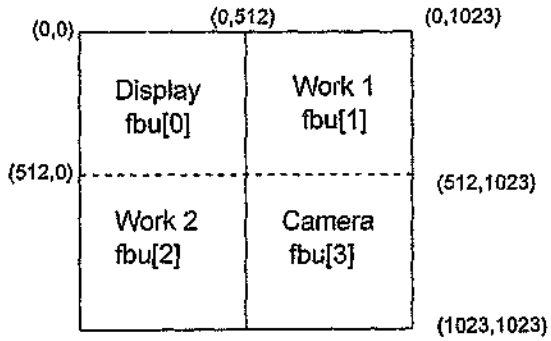


Figure 2-4. Memory Management

the computer program they are designated as four windows fbu[0], fbu[1], fbu[2] and fbu[3] respectively.

The camera window, fbu[3], is used to store the image snapped from the camera. The work 1 window, fbu[1], is used to preprocess the image. The image in the camera window is copied into this window. Thresholding and blob labeling operations are carried out on the data held in this window.

The work 2 window, fbu[2], is used to store blobs that are 'singled out'. Further operations and computations are undertaken for each individual blob in the **work1** window.

The display window fbu[0] works in a similar manner to a slide and is projected on to the image Display Buffer. Anything to be displayed will be first put into this window at a specific size and location.

A small window w[0] is defined to reside in the display window. It works in the same way as a lens, through which an image from the camera window is reduced by one half of its size. This window is a specific area designed for loading the image.

The image flow chart is shown in Figure 2-5

The rest of the frame buffer is used to load the object and blob templates. There are a total of 12 windows available.

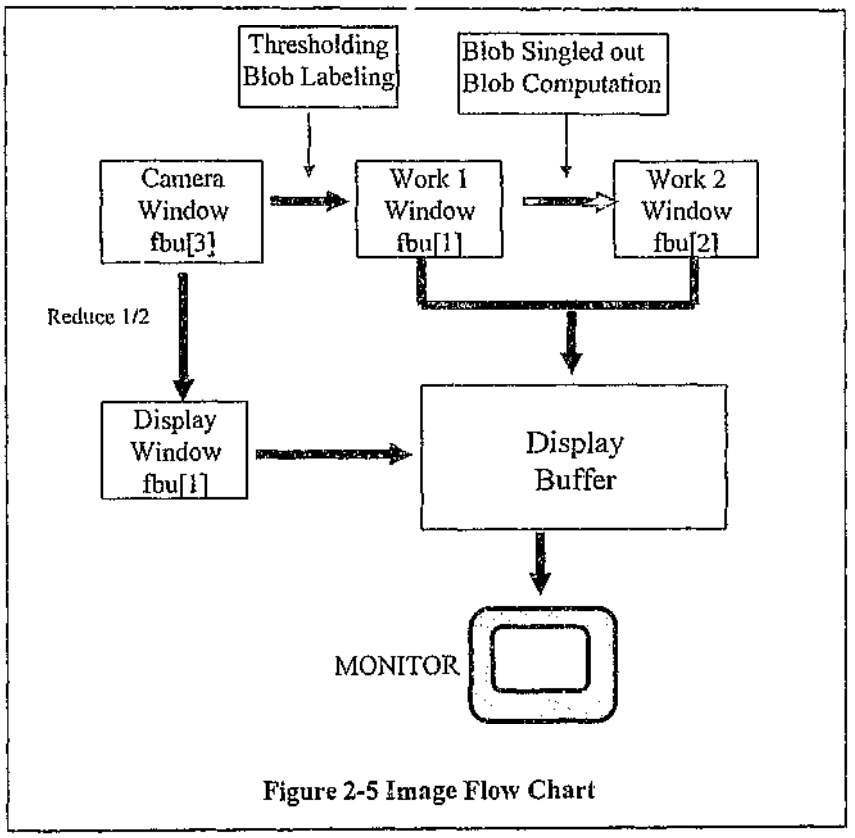


Figure 2-5 Image Flow Chart

2.7 Conventions Used in the Thesis

The conventions used in the thesis are defined from three aspects: origin, angle and direction.

2.7.1 Origin Convention

The origin convention for the frame buffer and all windows is the top left-hand corner. Therefore columns (horizontal) are y-direction, and rows (vertical) are x-direction. (see Figure 2-6).

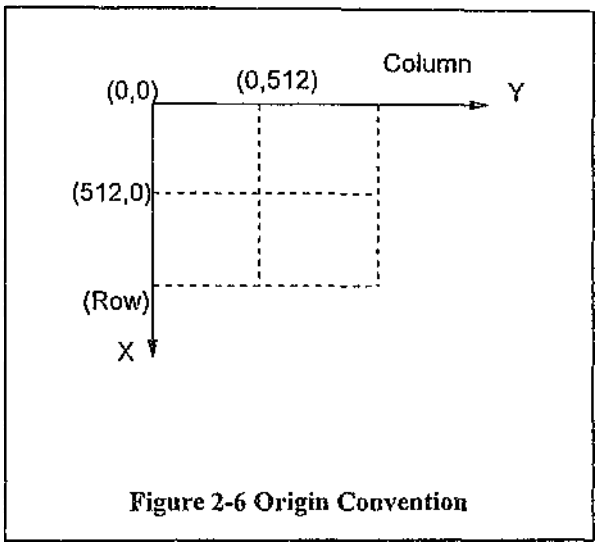


Figure 2-6 Origin Convention

2.7.2 Angle Convention

A positive angle is defined as an angle in which the direction of rotation from the x-axis is anti-clockwise as shown in Figure 2-7.

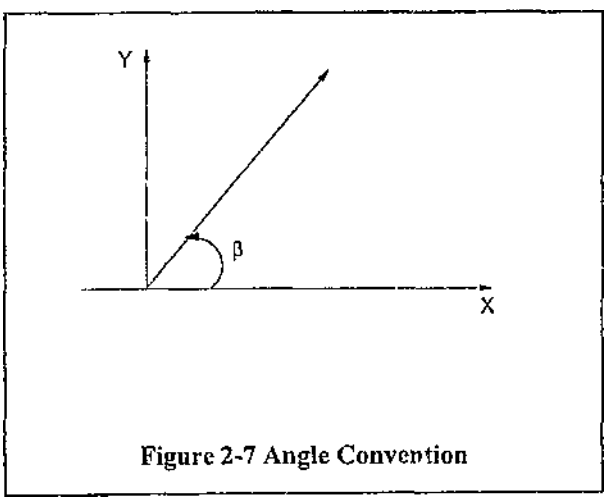


Figure 2-7 Angle Convention

2.7.3 Direction Convention

A blob is defined as a primitive chain. Each primitive is a vector, the direction of the primitive is always defined as being from the start point $P_1(x_1,y_2)$ to the end point $P_2(x_2,y_2)$. This convention is illustrated in Figure 2.8.

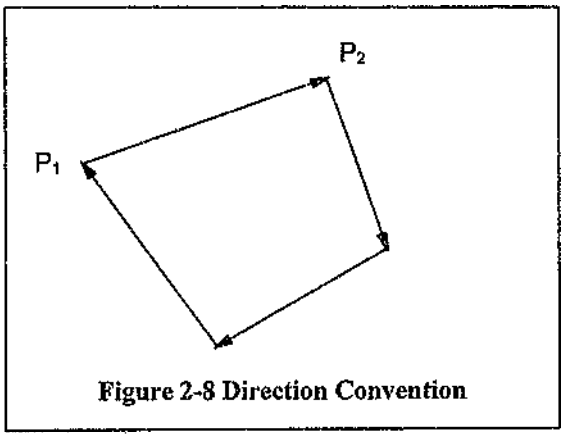


Figure 2-8 Direction Convention

3. IMAGE ACQUISITION AND USER INTERFACE

3.1 Image Acquisition

3.1.1 Application Domain

At present this system is restricted to the analysis of flat black mechanical parts on a white background.

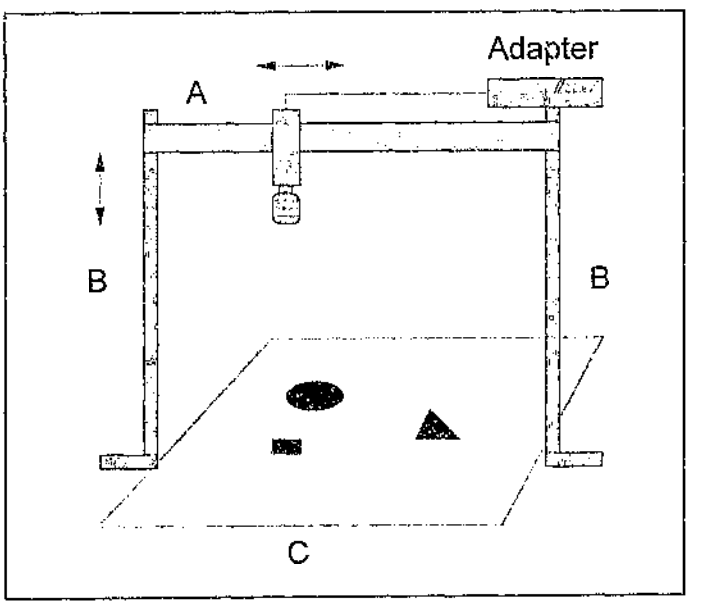


Figure 3-1 Image Acquisition Setup

No special lighting conditions are required. The set-up will operate satisfactorily in a room with ceiling mounted fluorescent light fittings, these conditions will provide sufficient contrast for the image.

The image acquisition setup is illustrated in Figure 3-1. The camera is positioned on a specially designed sliding track (A) supported with two vertical rails (B), which allow vertical movement of the track A. Movement in the horizontal and vertical directions is adjusted manually. It would be possible for the axes to be motorized for automatic relocation.

The image scene (C) is static at the present time, but it could be readily replaced by a conveyor belt when the application is developed to study moving objects.

3.1.2 Set up Parameters

Certain parameters associated with the image acquisition process must first be specified before any operations can take place. They are:

- the camera scan mode;
- the camera viewport, i.e. its location and size:
- the digitizer offset and gain.

The camera *scan mode* is set as external clock synchronization by configuring the Pi030 IAL function **Vxsnap** to type 2.

The camera *viewport* is a rectangular window of arbitrary size and location in the TV raster scan. It allows for the definition of an area of interest (AOI) in the

camera frame. Consequently, only the portion of the image inside the viewport will be digitized, thus saving acquisition time.

The camera *viewport* is defined by its origin coordinates Roff and Coff, and by its vertical and horizontal dimensions Nrow and Ncol respectively. These parameters are configured usi. the PICNIX console command **vr** to change the IAP registers. The digitizer *offset* and *gain* allows for the translation and scaling of the gray levels of the camera image. The modification of the translation and scaling parameters is implemented by invoking **vp** function in the PICNIX system.

These parameters are set up specifically for each application program and remain constant for all operations on this system.

3.2 Data handling and the Output of Results

During the process of image analysis, two types of data have to be handled; one set of data are displayed on the monitor and the other saved on disk. This data is required for the purpose of system surveillance and later data evaluation and processing and consists of images of the inspected scene and the numeric or charact. strings containing the processing results and/or error messages.

3.2.1 Displaying Images

Images are displayed on a Multisync colc. monitor. The procedure of displaying an image on the monitor can be described as following steps:

1. Initialize the display hardware (the display processor). This is implemented by calling the initialization function **Vdspinit** in the IAL function library. This function needs three key parameters to be specified:

   - A parameter to determine whether the "display device" is the frame buffer or the display buffer.
   - A parameter to decide the source of the image window to be displayed.
   - A parameter to determine whether the display is monochrome or color.

2. Image copying. When the display device is specified as the display buffer, the content of an image window is first needed to be copied into the display buffer. This is performed by the IAL function **Vdispl**. This function requires parameters to be specified regarding the location and the range of the image window to be displayed. The display buffer can be viewed as a specific window whose size is an entire frame of 512x512 pixels. Images exceeding one of the display buffer's boundaries are automatically clipped when they are copied into the display buffer.

3. Resolution reduction. This function is required when more than one frame of images are to be displayed in the same display frame or command menus are also displayed with the image. This function is implemented by applying the IAL function **Vreduc**. The function parameter *ratio* has to be specified to apply the required degree of reduction. In practice, a value of 2 for this ratio was applied for the work carried out on this project. Errors can occur when this function is applied to a 'non-solid' image. This is due to the fact that the

reduction is obtained by a coarse sampling, that is, only every 2nd, 3rd, 4th, ... pixel is taken, as indicated by the *ratio*. During the development and testing of the system, the only instances of a 'non-solid' occurring was when graphic models were used. The situation was corrected by specifying double pixel intensity to the drawing parameters, which in use were reduced by a factor of two when the reduction ratio was applied.

3.2.2 Storing Data on Hard Disk

There are no problems associated with the display of images on the Multisync color monitor. In addition the processing results and/or the error messages are displayed on the monitor of the host computer. However, in order to process, evaluate and analyze the results, both images and numeric information have to be stored on a disk.

For this purpose, a hard disk was purchased and connected via the Pi030 SCSI (Small Computer Systems Interface) controller using the following procedure:

(1) Set the hard disk parity to disabled;

(2) Set the hard disk ID to 1;

(3) Connect the power supply and the SCSI caole;

(4) Login the PICNIX system:

(a) Run command **cf scid** to change the configuration data on the small computer system interface as

Device name: **sd**;

Device ID (pun): 1;

(b) Reset the system by running the PICNIX command **rs**;

(c) Run SCSI 2 setup command **cfccs**;

(d) Format the hard disk by running the command **newfs sd**. A period of ten minutes is normally required before any message prompts appear on the monitor.

The above hard disk installation procedure is described in detail in order to assist subsequent users to avoid the faults which were observed during this operation.

The IAL function library does not provide utilities to implement the process of data saving on the hard disk. Special routines have been developed to implement the functions of data saving in the form of stream files and data reading from the stream files.

The data files are transferred to the host computer hard disk through the Ethernet by the remote copy functions provided by the Linux system.

3.3 User Interface

While the system program is running, the user needs to communicate with the system interactively: issuing commands and specifying parameters.

This system uses a command menu to implement the first requirement and a specific routine to input numeric or character data from the host computer keyboard.

The command menu is built by applying two IAL functions: **Gcursor** and **pinside**.

The function **Gcursor** is used to generate a cursor and move it by the mouse into any desired position in the frame buffer. The positioniug procedure is terminated by a click, on which event the function returns the cursor coordinates.

The function **pinside** is applied to test whether a point is inside a specific window.

The buttons in the command menu are defined as a sequence of windows in the frame buffer fbu[0]. A command is activated once the mouse is positioned and clicked within a button window. Figure 3-2 illustrates the structure of the command menu.

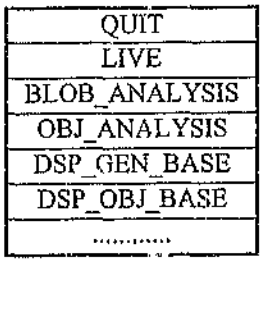


Figure 3-2 Command Menu

The routine used to input data from host keyboard is developed by a system call function **read(d,buffer,nbytes)**. Special attention is required to be given on the

definition of the function parameters. The object referenced by **d** in the read string is set equal to 0 in this procedure.

4. HIGH LEVEL IMAGE SEGMENTATION

4.1 Introduction

When a human observer views a scene, the neurological processing that takes place in the retina and optic cortex essentially segments the scene for the observer. This is done so effectively that the perception is not of a complex scene, but rather a collection of objects.

In a machine vision system, under the conditions being considered here, the input is an image scene of industrial parts. For the machine to understand the image scene, a description of the given scene is required. A picture description can be thought of as a relational structure that involves parts of the picture, properties of those parts, and relationships among them. Given such a structure, it becomes relatively straightforward to answer questions about the picture, or to generate statements about it. In particular, it is easy to determine whether the picture contains specified arrangements of parts, i. e., whether the structure contains specified substructures. Therefore, the purpose of segmenting the image into specific parts is the correct procedure for generating such a description.

The image segmentation task is tackled at two levels: at the high level of image segmentation, the system deals with the problem of extracting regions or objects from the image and computation of properties of objects and relationships among the objects. At the low level of image segmentation, the objects or blobs are decomposed further into basic primitives. The descriptions of the objects or blobs

involve their primitives and connection types between the primitives. This topic is the subject of discussion in the next chapter.

In this chapter, only the high level image segmentation is considered. First, image segmentation techniques are reviewed as a framework with which to establish an appropriate segmentation approach. The operations needed to accomplish the segmentation goals are discussed and the method to implement them is proposed. The question of how to compute an object's features is also discussed, and finally, the problems in programming are discussed by considering and analyzing some experimental results.

4.2 A Brief Review of the Segmentation Techniques

Many techniques and theories on segmentation have been developed during last three decades. They can be broadly classified as Thresholding, Edge Detection, Matching and Tracking.

4.2.1 Thresholding

The most common way to extract objects from a background is by **Thresholding**. Thresholding classifies the pixels of a given image into two groups, that is, objects and background. The early example was given as locally averaging and then thresholding for the textured region extraction (Narasimhan, 1963).

The crucial problem in using thresholding technique is that of how to select the threshold. Doyle (1962) suggested the p-tile method of threshold selection, in this

method an approximate p-tile of the distribution is used as a threshold when the approximate area of the desired objects is known.

The p-tile method is one way of using a picture grayscale histogram as a guide in the threshold selection. Another approach that made use of the histogram is called the Mode method, this method was proposed by Prewitt and Mendelsohn(1966). If the objects are clearly distinguishable from the background, the grayscale histogram will be bimodal and the threshold for the segmentation can be easily chosen as being the value at the bottom of the valley between the two modes.

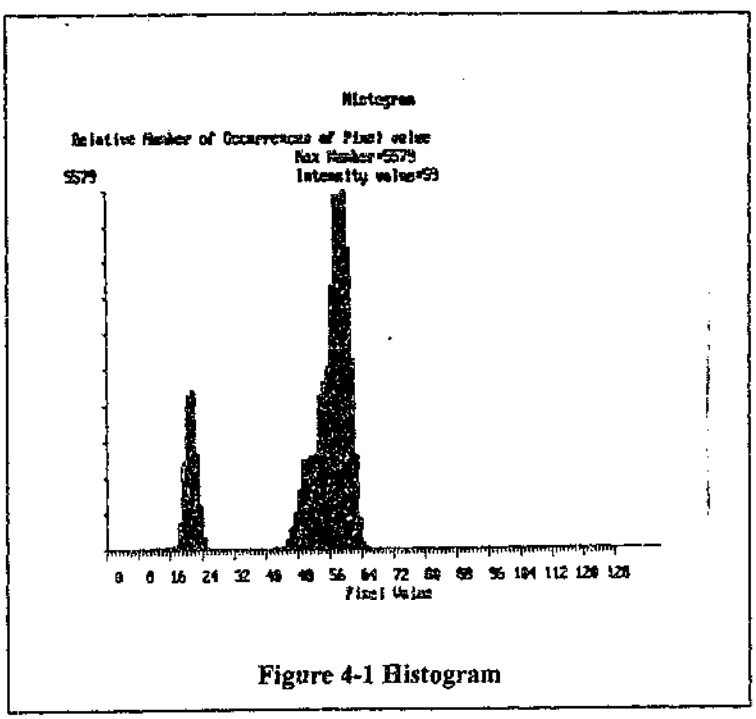


Figure 4-1 Histogram

Figure 4-1 is the histogram of black object superimposed on a white background.

Cheng and Don (1991) proposed a morphological approach to solve the problem when the grayscale histograms are not bimodal.

4.2.2 Edge detection

Another important approach to picture segmentation is based on the detection of discontinuity, i.e., of places where there is a more or less abrupt change in gray scale or in texture, indicating the end of one region and the beginning of another. Such a discontinuity is called an Edge. This approach is called the Edge Detection approach. It aims to find the object boundaries directly by their high gradient magnitudes.

Derivative operators, which give high values at points where the grayscale of the picture is changing rapidly, evidently can be used as an edge detector. The use of derivative operators, particularly the gradient, for edge detection dates back to Kovasznay and Joseph (1953). As most image processing system dealt with digitized pictures, many researchers have proposed a digital approximation for these operators.

The simplest derivative operators are the first partial derivatives $\partial f/\partial x$ and $\partial f/\partial y$, which give the rates of change of gray level in the x- and y-directions. The computer can carry out many mathematical operations, but unfortunately differentiation is not one of those operations. Therefore, derivatives must be replaced by finite differences. For the digital image, the finite differences are defined as follows:

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

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

where $f(i,j)$ is the input image in units of intensity and with integer pixel coordinate (i,j).

Roberts(1965) proposed a very popular digital gradient approximation:

$$\{[\sqrt{f(i,j)} - \sqrt{f(i+1,j+1)}]^2 + [\sqrt{f(i+1,j)} - \sqrt{f(i,j+1)}]^2\}^{1/2}$$

The inner square roots are used to make the operation resemble the processing that takes place in the human visual system.

Higher-order derivative operators can also be used to detect edges in an image. The Laplacian ($\partial^2 f/\partial x^2$)+($\partial^2 f/\partial y^2$) is such an operator that is orientation insensitive. In the digital case, it is approximated by the sum of the vertical and horizontal second difference:

$$\nabla^2 f(i,j) = f(i+1,j) + f(i-1,j) + f(i,j+1) + f(i,j-1) - 4f(i,j)$$

This can be regarded as a convolution with the kernel as shown below:

0	1	0
1	-4	1
0	1	0

Figure 4-2 Laplacian Operator

Other methods that can be applied to the edge detection problem are best-fitting functions, and finding edge by using planning (Kelly,1971).

4.2.3 Matching

Picture Matching, also called template matching, is an approach where points are found at which an arbitrary given pattern of grayscale is present. There are many possible ways of measuring the degree of match. The generally accepted measure is Cross Correlation. For digital image processing, this generally involves convoluting an image with one or more masks, followed by simple nonlinear processing.

In a convolution, the center pixel of a neighbor is replaced by a new value obtained by first multiplying first all the pixels of the neighborhood by a coefficient called a weight factor and then summing up all these products. The matrix of weight factors for a convolution is called its kernel.

The operations of convolution act as spatial-frequency filters for removing or enhancing selective frequency components found in an image. Much work has been done in designing various masks, such as lines, curve, edge streak, etc.(You and Cohen,1991)

4.2.4 Tracking

The methods described in the previous section are regarded as parallel segmentation methods in which the processing did not depend on results already obtained at other points.

Tracking is a method which processes a point by taking advantage of the results at previously processed points. This sequential segmentation method has a potential advantage over parallel methods, with respect to its computational cost on a

sequential computer. Raster tracking and region growing are traditional techniques.

4.3 Selection of Segmentation Method

Even though many segmentation techniques have been developed, most ad hoc methods are very application specific. Thresholding as the oldest segmentation technique is particularly suitable for scenes containing a solid object resting upon a contrasting background. It is computationally simple and never fails to define disjointed regions with closed connected boundaries.

In view of the fact that the majority of parts presente for identification in a manufacturing environment will be regular, flat, mechanical parts, thresholding was chosen as the segmentation technique for the system being developed.

The first task of a thresholding technique is to compute the **histogram** of the image. A **histogram** is a graph of the frequency distributions of the pixel-intensity values for an image or portion of an image. It can indicate a great deal of information about the overall brightness and contrast of an image. The dynamic range of the pixel values that make up an image are readily apparent. Therefore, it is a valuable tool for image processing both qualitatively and quantitatively, these features are apparent from a study of Figure 4-1.

The most important role of the histogram is to indicate the appropriate threshold setting for image segmentation. For this purpose, the histogram must be analyzed to find the number of peaks, their height, and their location. This is the second task in the thresholding technique.

Ideally, a histogram has a bimodal distribution, i.e., two peaks, as illustrated in Figure 4-1. If it is assumed that the highest peak in this histogram indicates the grey levels of the objects of interest, then the second highest peak indicates the average gray levels of the background. In this case, the appropriate threshold selection for separating the objects from the background is the mid-point in the valley between the two peaks. If the distribution is not bimodal, image enhancement operations are required. In the environment being considered, the histogram normally has a bimodal distribution.

A human operator has no problem in identifying the discontinuity between the two modes, however, for the machine to automatically select the appropriate threshold value, a special algorithm is needed.

It is proposed that the Minimum-Error Thresholding principle is applied to the analysis of the histogram program. The following text illustrates the application of this principle.

Let the image consists of objects on a background, where the gray levels of points in the objects have a normal probability density p(z) with mean μ_1 and standard deviation σ_1, while the gray levels of background points have a normal density q(z) with mean μ_2 and standard deviation σ_2, ($\mu_1 < \mu_2$). Suppose further that the objects occupy fraction θ of the image area, so that the background occupies fraction 1-θ.

$$I = \theta p(z) + (1-\theta)q(z). \qquad (4.1)$$

where I is the image overall gray level probability.

If σ1=σ2=σ, the threshold that minimizes misclassification probability is :

$$t = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_2 - \mu_1} \times \ln \frac{\theta}{1-\theta} \qquad (4.2)$$

To prove this principle, let the image be thresholded at t, then all points with a gray level <t are defined as object points, and all points with level ≥t as background points. The probability of misclassifying an actual background point as an object point is

$$Q(t) \equiv \int_{-\infty}^{t} q(z)dz \qquad (4.3)$$

Similarly, the probability of misclassifying an object point as background is

$$1 - P(t) \equiv \int_{t}^{\infty} p(z)dz \qquad (4.4)$$

The overall misclassification probability when the threshold is set at t is thus:

$$\theta(1\text{-}P(t)) + (1\text{-}T)Q(t) \qquad (4.5)$$

To find the threshold value for which the probability of misclassification is a minimum, this expression is differentiated with respect to t and set the result equal to zero, obtaining:

$$(1\text{-}\theta)q(t) = \theta p(t) \qquad (4.6)$$

Now

$$p(t) = \frac{1}{\sqrt{2\pi}\sigma_1} \exp[\frac{-(t-\mu_1)^2}{2\sigma_1^2}] \qquad (4.7)$$

$$q(t) = \frac{1}{\sqrt{2\pi}\sigma_2} \exp[\frac{-(t-\mu_2)^2}{2\sigma_2^2}] \qquad (4.8)$$

If expression (4.7) and (4.8) are substituted in the previous equation (4.6) and then taking logarithms of both sides, the following result is obtained:

$$\ln\sigma_1 + \ln(1-\theta) - \frac{(t-\mu_2)^2}{2\sigma_2^2} = \ln\sigma_2 + \ln\theta - \frac{(t-\mu_1)^2}{2\sigma_1^2} \qquad (4.9)$$

If $\sigma_1 = \sigma_2 = \sigma$ then

$$\sigma^2(t-\mu_1)^2 - \sigma^2(t-\mu_2)^2 \approx 2\sigma^2\sigma^2 \ln\frac{\theta}{1-\theta} \qquad (4.10)$$

Then

$$t = \frac{\mu_1+\mu_2}{2} + \frac{\sigma^2}{\mu_1+\mu_2}\ln\frac{\theta}{1-\theta} \qquad (4.11)$$

Suppose the histogram of the image has a mean μ and a standard deviation τ, then formula (4.11) can be expressed as

$$t = \mu + (p_1 \times \tau)/100 \qquad (4.12)$$

where $\mu = \mu 1 + \mu 2$; and

p_1 is a parameter related to the practical application that can be obtained from experimental results.

When the σ_1 is approximately the same as σ_2 the formula has a similar form:

$$t = \mu + (p_1 \times \tau)/100 + p_2 \qquad (4.13)$$

The parameter p_1 and p_2 can be either positive or negative, depending on whether the object is bright and the background dark, or vice versa. In the first case where, p_1 and p_2 are negative, the threshold is lower than the mean. In the second case

where, p_1 and p_2 are positive, the threshold is higher than the mean. In both cases it is assumed the mean will represent the pixels of the object of interest.

The stage of the thresholding scheme can be summarized as follows:

(1) Compute the image histogram,

(2) Analyze the histogram and compute its mean μ and standard deviation τ,

(3) with known parameters p_1 and p_2 (obtained experimentally), use formula (4.13) to compute the optimal threshold value.

As an alternative to the above methodology it is possible to obtain the threshold value automatically by the application of a peak searching function to the histogram. Referring to Figure 4-1, it can be seen that the threshold setting is approximately at the midpoint between the two peaks, i.e.:

$$t = (Peak1 + Peak2)/2; \qquad (4.14)$$

Although it produces an error if the midpoint does not exactly coincide with the minimum of the valley, it is still a good approximation.

4.4 Image Structure

Once the objects are extracted from the image, further measurements and processing on the individual object are still required. Therefore, it is necessary to extract and store the individual objects in a convenient format.

4 HIGH LEVEL IMAGE SEGMENTATION

There are three popular ways to structure the image. They are boundary chain encoding, runlength encoding and membership mapping.

Run-length encoding is a one-dimensional data-compression technique that is popular in PC-based graphics packages. The amount of compression achieved by the run-length encoding is image file dependent but typically reduces the storage requirement by one half to three quarters.

Run-length encoding stores sequences of related values. The number of repetitions of a particular value is stored along with the pixel values that are repeated. If pixels in a horizontal line are all different, the data-compression technique can be a disadvantage. It is possible and popular to use hybrid procedures that incorporate both techniques: run-length encoding scheme or a direct scheme.

One important factor leading to extensive application of run-length encoding method is that it reflects one of the geometrical properties of extracted objects, that of "connectedness". It is also obvious that an approach which only tests the start and end run-length values will require less operations than a pixel masking technique.

Even though Run Length Encoding is a good way of storing an image, it would appear to be inconvenient for image processing operations.

In the machine vision system being developed, image processing performs mathematical operations on two-dimensional images such as an entire image or on a window within an image. This means that a processing buffer is needed and the run length encoding will make the image processing operation on an object difficult.

3	2	1
4	p	0
5	6	7

Figure 4-3 Chain Encoding

Boundary chain encoding was first proposed by Freeman(1961). It describes the segmented object by its boundary. It is based on the fact that boundaries are connected paths. The chain encoding starts by specifying the x,y coordinates of an arbitrarily selected starting point on the boundary of the object. That pixel p has eight neighbors, and at least one of these must also be a boundary point. The chain encoding specifies the direction in which a step must be taken to go from the present to the next boundary point. Since there are eight possible

directions, they can be numbered, say, from 0 through 7. Figure 4-3 shows a possible assignment of the eight direction codes. The boundary chain code then consists of the coordinates of the starting point, followed by the sequence of direction codes that specify the path around the boundary.

With the boundary chain encoding, the segmentation of an object requires, for storage, only one x,y coordinate and three bits for each boundary point. The chain code is invariant to the translation of the object, while a rotation results in a circular shift of the string of integers. Even though it is a more compact format for storing image information, it is not useful in the system environment being developed as further processing of the individual objects is required.

The object membership map is an approach that stores the segmentation by generating a separate image, the same size as the original, and encoding object

membership on a pixel-by-pixel basis. In the object membership map, the gray level of each pixel encodes the sequence number of the object to which the corresponding pixel in the original image belongs.

The advantage of the object membership map is that it allows all of the object to be processed in sequence and makes it more convenient to carry out further measurements on each individual object.

The disadvantage of object mapping is that it is not a particularly compact way to store the segmentation information. However, considering the system configuration being used is furnished with fast image processing hardware and large space memory buffer, this problem is not critical.

After reviewing the alternatives for segmentation it was decided to employ the **Blob Labeling** method, which is similar in concept to the object membership map, as the high level image segmentation procedure to be used in this project.

The first step of blob labeling is to give each blob in a selected image window a number (label) thus all the blobs in the image window can be further processed by specifying their labels. The individual blobs being singled out by blob labeling, can be readily subjected to further feature analysis.

The second task of blob labeling is to build a tree relationship between the blobs that are singled out. In an image each pixel can belong to only one blob; however blobs may have internal holes, in which case these holes are defined as separate blobs.

The method of building the blob relationship is governed by the following rules:

Given any two sets of connected pixels (blobs), say blob A and blob B, then these blobs obey the following rule A and B either are totally disjoint, i e. their intersection is the empty set, or A is totally included in B, or B is totally included in A. In mathematics, this property is called a nested set property.

In the tree structure representing the nested set relationship among the blobs, all the blobs that are at the same level of the tree are "siblings", and a blob that is nested in another blob is the other blob's "child", while the other blob is the

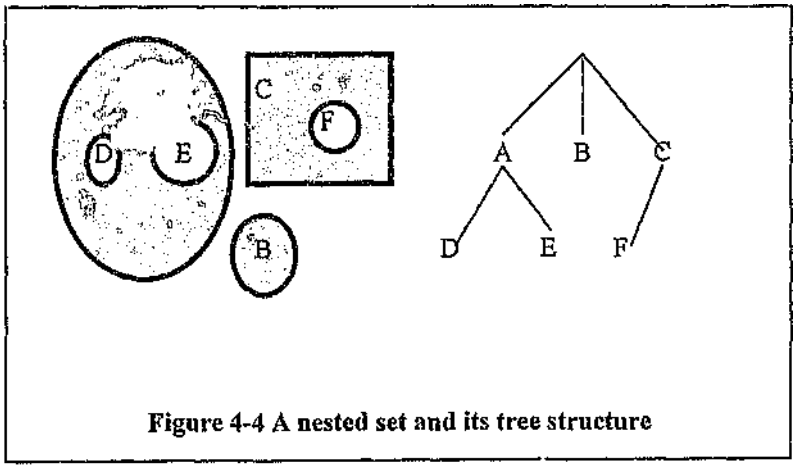


Figure 4-4 A nested set and its tree structure

"parent" of the one that is nested.

4.5 Feature Extraction and Computation

This section discusses the problems of Feature Extraction and Feature Measurement. Feature Extraction defines what is to be measured, and Feature Measurement details on how those measurements are accomplished. The section

begins with a brief literature review of the Feature Extraction technique, followed by an analysis of the descriptive features of this system.

4.5.1 A Brief Review of Feature Extraction

The purpose of feature extraction is to obtain a set of data upon which a subsequent recognition process can be based. Appropriate feature selection then becomes the key to successful recognition.

Significant attention was given to the area of feature extraction in early 70's. A special issue on feature extraction was devoted by IEEE transaction on Computers in 1971.

The features selection techniques that were proposed can be divided into the following three categories:

1) Statistical,

2) Information theoretic,

3) Sequential.

Mucciardi and Gose (1971) presented seven techniques for choosing good subsets of features and compared their performance on a nine-class vector cardiogram classification problem.

Most of the feature extraction literature has centered around finding linear transformations, which maps the raw measurements into a smaller set of features which hopefully contain all the relevant or discriminating information. The most popular transform is the Karhunen-loeve or eigen vector orthonormal expansion.

Foley and Sammon (1975) suggested and derived an algorithm for extracting features for a two-class problem in which the criteria for selecting each feature is based directly on its discriminatory potential.

Shen and Pilkey (1991) gave a good survey of existing techniques of feature selection, and proposed a ranking scheme for features selection based on a feature's calculated "performance potential".

4.5.2 Selection and Computation of the Features

The decision on how to choose the method of feature extraction in machine vision system is heavily dependent upon the recognition scheme employed at the recognition stage. This point will be explained in detail in Section 6.2. The feature extraction method used in this project is based on the two levels of image segmentation operations. At the high level of the image segmentation, blobs which are labeled and singled out are described by global features. During the low level image segmentation stage, only local features on the blobs are involved. This will be discussed in the next chapter.

In this section, the discussion is limited to the computation of the global features on the blobs. The global features are classified into two categories:

1) **Geometric Features** - features based on contour and shape of the blobs being singled out.

2) **Relationship Features** - features based on the nested set relationship among the blobs.

4.5.2.1 Geometric Features

The range of geometric features that can be measured on the individual blobs are area, gravity centroid, perimeter, moments, compactness ratio and thinness ratio.

Area: The blob's **area** is taken by simply counting the number of pixels in the blob.

Perimeter: Measuring the **perimeter** can be achieved by counting the pixels in the boundary of the **object**. However, this ignores a number of difficulties - in particular, that the distance between horizontal and vertical pixels is 1 but the distance between diagonal pixels is roughly 15% larger. This situation is illustrated in Figure 4-5.

An improved estimate of the perimeter was used by counting the number of pixels in the P_8 and P_4 perimeters. The product obtained is a good approximation of the square of the perimeter (P^2.).

P_4 - represents the object perimeter taken by the four-neighbor-connected

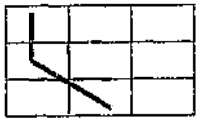


Figure 4-5 Distance Between Diagonal Pixels

definition.

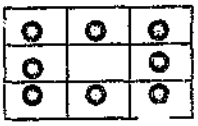
P_8 - represents the object perimeter taken by the eight-neighbor-connected definition.

The 8-neighbor-connected and 4-neighbor-connected are two types of definition which describe the object boundary connectedness. The difference between the two definitions is the definition used for a pixel's immediate neighbors. In the 8-neighbor-connected definition, a pixel is defined as being connected to its eight nearest neighbors. In the 4-neighbor-connected, it is defined to, has four horizontal and vertical neighbors. These definition are shown in Figure 4-6.

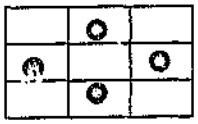
P_8 and P_4 were obtained by using Boolean Operator for boundary points to operate on the captured image. Pixels in image processing problem are usually described as a 3x3 arrangement numbered as shown in the Figure 4-7.

a_1	a_2	a_3
a_8	a_0	a_4
a_7	a_6	a_5

Figure 4-7 Pixel Neighbors



8-connected



4-connected

Figure 4-6 Neighbor Definition

A boundary point is regarded as a foreground point pixel that has at least one neighbor that is a background pixel. The condition for at least one 8-connected neighbor being a background pixel can be written:

$$B = (NOT\ a_1)OR(NOT\ a_2)OR(NOT\ a_3)OR(NOT\ a_4)OR(NOT\ a_5)$$

$$OR(NOT\ a_6)OR(NOT\ a_7)OR(NOT\ a_8)$$

Similarly the condition for at least one 4-connected neighbor being a background pixel can be written:

$$B = (NOT\ a_2)OR(NOT\ a_4)OR(NOT\ a_6)OR(NOT\ a_8)$$

The final local Boolean Operator is

$$a_0\ AND\ B$$

Therefore, the operators for P_8 and P_4 both contain two parts, one which detects a certain configuration of a_1 to a_8, and another which combines this with the original values of the image a_0.

Centroid: The blob's **centroid** is the center of the area. It is used as a representative point of the position of the blob. The center of the area is the center of the mass of a figure of the same shape with constant mass per unit area. So it is also called the gravity centroid of the object.

Calculation of the centroid is based on the definition in Physics: the center of mass is that point where all the mass of the object could be concentrated without

changing the first momer . of the object about any . xis. As our problem domain is constrained by the two-dimensional situation, the moment about the x-axis is

$$x_0 \iint_I b(x,y)dxdy = \iint_I xb(x,y)dxdy,$$

and the moment about the y-axis is

$$y_0 \iint_I b(x,y)dxdy = \iint_I yb(x,y)dxdy,$$

where $(x_0\ y_0)$ is the position of the center of area, b(x,y) is the characteristic function, the integration is over the whole image I. The integrals appearing on the left of these equations are the area A, i.e. that A is the zeroth moment of b(x,y).

Then we get

$$x_0 = [\iint_I xb(x,y)dxdy]/A,$$

$$y_0 = [\iint_I yb(x,y)dxdy]/A,$$

As the image has been thresholded into the binary image, the computation of the centroid or the first moments can be simplified.

Suppose b_{ij} is the value of the binary image at the point in the i^{th} row and the j^{th} column, where the image window imposed on the image buffer is m×n. The algorithm of calculation of the centroid is as follows:

From i=1 to m and j=1 to n

check if $b_{ij} = 1$,

if yes, add i, j to the accumulated totals for the first moment,

if no, continue the above checking on next i and j.

The **orientation** of the object is defined by the orientation of the axis of elongation. It is the axis of least **second moment**. It will not be used as a global feature, since it is too crude a measurement for a blob's orientation.

Compactness ratio: The **Compactness ratio** (P^2/A), the perimeter squared divided by the area of the object, is the best known shape index. This quantity is independent of changes in scale and rotation.

4.5.2.2 Relationship Features

The relationship features used in the system to represent the nested set are four members: the right-hand sibling, the left-hand sibling, the leftmost child, and the parent, respectively. It is a tree structure, as illustrated in Figure 4-4. It shows that they are two doubly linked lists representing both the sibling-sibling and the parent-child relationships. From those lists all questions concerning the nested set relation can be answered.

In the example in the Figure 4-3, A has B as right-hand sibling (in turn, B has C as right-hand sibling). The leftmost child of A is D (E is also a child of A, however, it can be reached only as a sibling of D).

4.6 Implementation

The procedure of the high level image segmentation is illustrated using a flow chart in Figure 4-8.

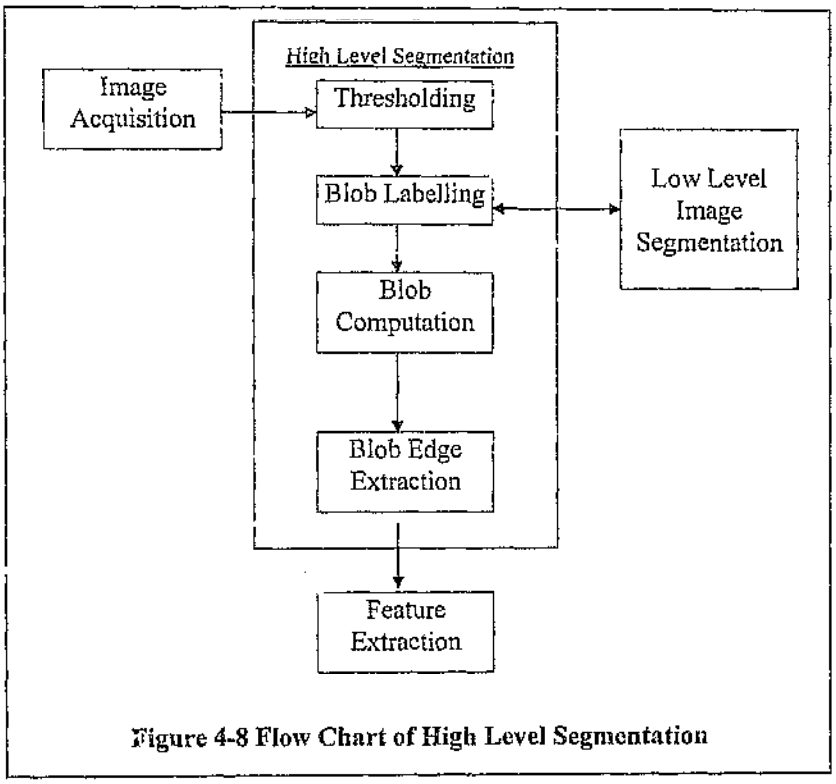


Figure 4-8 Flow Chart of High Level Segmentation

The input to the high level segmentation module is the result from the Image Acquisition Module. The outputs go to the Low Level Image Segmentation Module and the Feature Extraction Module.

The aims of this module are to extract the object from the captured image, segment it into blobs, find the edge of each blob and compute the global features on each blob. It consists of four major routines: Threshold, Blob_label, Blob_compute and Blob_edge routines.

Thresholding routine: The input is a frame of grayscale image in the camera buffer fbu[3]. The routine starts by computing the histogram of the image, the result of which is a binarized image.

First, the image frame in the camera buffer is transferred into the working buffer fbu[1]. The histogram is taken over the working buffer fbu[1]. The results are stored in a buffer variable histogram[256], which is a one dimensional integer array, in which the entries are corresponding to the grayscale values of 0 to 255.

The requirement of the auto thresholding scheme, which was discussed in section 4.2.1. is to set up the appropriate parameters p1 and p2. Experiments carried out using typical mechanical parts, yielded the values of 80 for p1, and 0 for p2.

After thresholding, the image in the working buffer fbu[1] is converted to a binarized image.

Blob labeling routines: The blob labeling operation is carried out on the binary image in the buffer fbu[1] and the result is stored in the same buffer.

The blob labeling is performed by the IAL function Vblob. Vblob can label up 128 white blobs and up to 128 dark blobs occurring in a source window.

The results of blob labeling are: number of white blobs (Nwhite), number of the black blobs (Nblack), and two sets of consecutive labels. The even labels 0, 2, 4, ..., are used for the black blobs and the odd labels 1, 3, 5, ..., are used for the white blobs.

After the blob labeling has been completed, , the tree building function Vbbtree in IAL function library is used to performing the task of building the tree structure representing the nested set relationship among the blobs. Four pointers are provided in the program to implement the procedure:

btree[i][0] = pointer to the right-hand sibling of blob labeled i,

btree[i][1] = pointer to the left-hand sibling of blob labeled i,

btree[i][2] = pointer to the leftmost-child of blob labeled i,

btree[i][3] = pointer to the parent of blob labeled i,

The result of the blob labeling routines is a frame containing a labeled image in the buffer fbu[1] and the tree structure data in the form of a short array btree[256][4].

Blob computation routines: The procedure of the blob computation routine procedures are carried out using the following steps.

(1) Filter out the individual blobs in sequence and paste them on the working buffer fbu[2]. Small blobs whose areas are smaller than a given value are ignored.

(2) Impose a bounding box for each singled out blob. This is then defined as a window, This operation clips the image buffer into a smaller area and decreases the computation burden. In next chapter, it will be shown that this operation also plays an important role in searching for dominant points.

(3) Compute the area and centroid for each individual blob.

(4) Compute the P_4 and P_8 perimeters by applying the Boolean operators, described in Section 4.5.2.1 to every pixel in the bounding box window. The perimeter P and compactness ratio can be calculated from P_4 and P_8 perimeters.

Blob edge extraction: The blob Edge is one of the most important features by which a shape is described. When further operations and computations are needed to be performed on the extracted blob, using the edge extraction routine will save significant amounts of pixel searching and calculation time.

As explained in Section 4.2.2, a considerable amount of work has been done on the design of edge filters. These methods can be thought as *neighbourhood operations*, which compute for every pixel a new value as a function of the grey levels of the pixels of a surrounding neighborhood including the pixel itself. The computer implements these neighborhood operations by *convolution*. The pixel transformation is the weighted sum of the pixels values of the neighborhood. The weights are called the coefficients or the kernel of the convolution.

The Pi030 machine used for this work, provides square neighborhoods which are either 3x3, 5x5, or 7x7 pixels large. The user can define a specific convolution by specifying one of these coefficient matrices.

In this project the Laplacian 5x5 operator has been chosen to extract the boundary

0	0	-1	0	0
0	0	-1	0	0
-1	-1	8	-1	-1
0	0	-1	0	0
0	0	-1	0	0

Figure 4-9 Laplacian 5x5 Kernel

of the blob. The convolution kernel is:

The Laplacian Cross is a special case of a square neighborhood with the coefficients in the diagonal fields being zero. The Laplacian operator enhances the high spatial frequencies in the image which represent the edge information, i.e. it acts as a high-pass filter. High-pass filtering tends to enhance the noise. Therefore to reduce the noise sensitivity of the Laplacian operator, some researchers have suggested that it is combined with an averaging procedure which serves to reduce the noise.

Experiments have shown that this system works well with the Laplacian 5x5 operator and the results are presented in Figure 4-10 to Figure 4-14.

4.7 Experimental Results

Comparison of Different Edge Operators:

Figure 4-10 and Figure 4-11 are two edges extracted by taking the Horizontal Emphasis operator and the Vertical Emphasis operator, respectively, on the same

image. They show that the edges are fuzzy in the corresponding horizontal and vertical directions.

Figures 4-12 to Figure 4-14 are edges extracted by the Laplacian Cross 3x3, 5x5 and 7x7 operators. The results show that the edge becomes thicker as the convolution kernel increases. It is evident that the Laplacian Cross 5x5 is an optimal option.

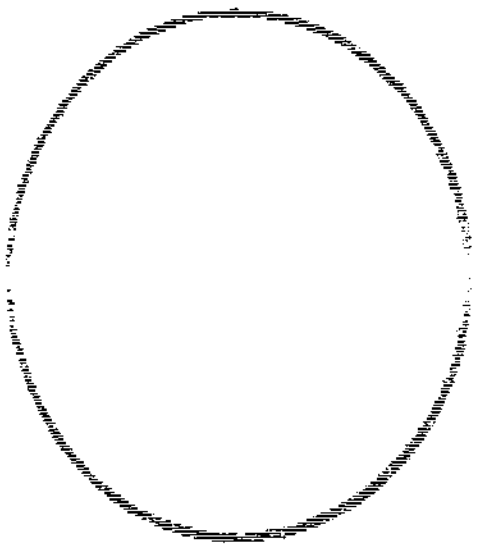


Figure 4-10 Horizontal Emphasis Operator

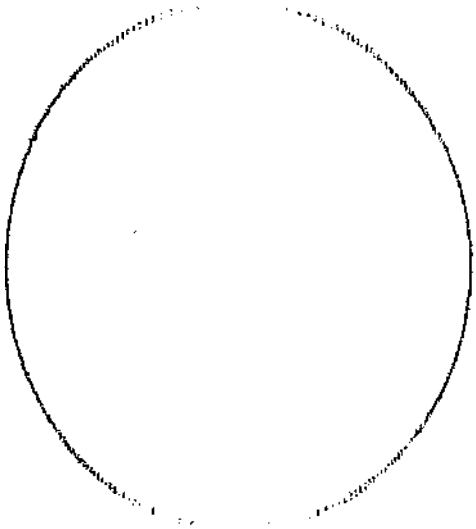


Figure 4-11 Vertical Emphasis Operator

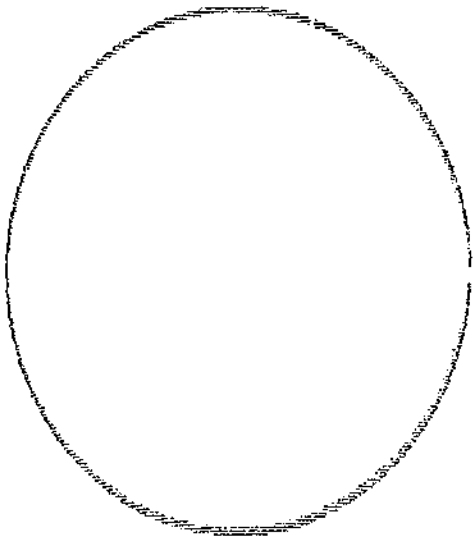


Figure 4-12 Laplacian Cross 3x3 Operator

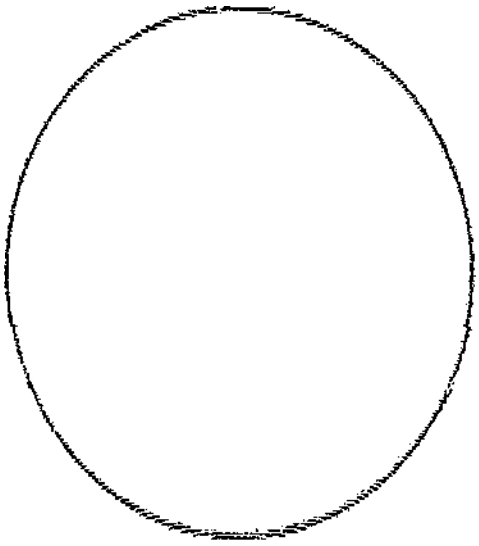


Figure 4-13 Laplacian Cross 5x5 Operator

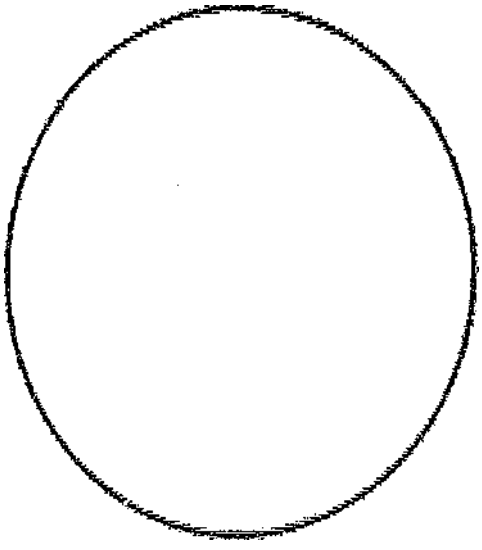


Figure 4-14 Laplacian Cross 7x7 Operator

Labeling Blobs:

Figure 4-15 illustrates an object processed by the Blob Labeling function. Each blob is singled out with a bounding box.

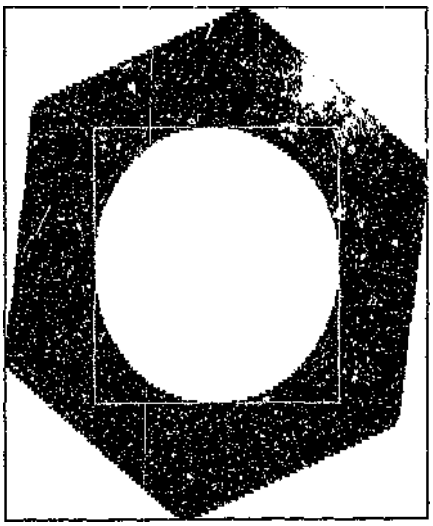


Figure 4-15 Labeling Blobs

Thresholding and Computation:

Figure 4-16 is the image of a thresholded blob, with a centroid illustrated by a cross.

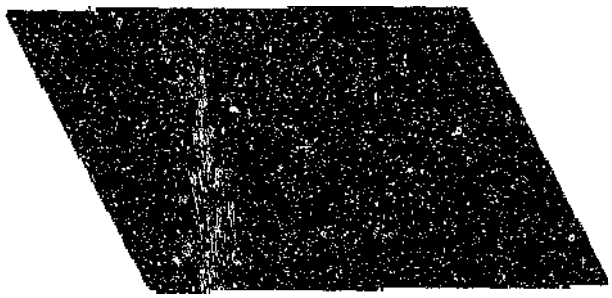


Figure 4-16 Thresholding Blob

The features are extracted by Blob Computation function as follows in the data file, pl.dat:

Blob features are extracted as
Numbers of Lines: 4
Numbers of ARCs: 0
Numbers of Lines with equal length: 2
Numbers of equal angles: 2
Numbers of Angles =90 : 0
Numbers of Angles >90 : 2
Numbers of Angles <90 : 2
Area = 46922
Gravity Center (174 , 247)
Perimeter = 681
Compactness Ratio = 9.889498
Thinness Ratio = 1.270678

5. LOW LEVEL IMAGE SEGMENTATION

5.1 Introduction

In the previous chapter, the techniques used to extract blobs and their edges from the given image and the sequence of numbers assigned to each individual blob was explained. Once the blobs and edges have been extracted, the next stage of the procedure was to carry out the computation of features and determine the relationship features for each blob. This stage was defined as high level image segmentation.

This chapter deals with extraction and computation of local features on each blob. The global features, relationship features and the local features combine to give a description which enables the machine vision system to perceive what it sees. The question of how to design a good shape description scheme and how to use it for the machine to understand the image will be discussed in next chapter under the heading of "Building the Knowledge Based System".

The focus in this chapter is on the description at the blob level. It is based on the idea that the outline of the blob can be used to represent the blob and it can be segmented into lines and curves. This has turned out to be regarded as 'non trivial' work by many researchers. Even though a lot of algorithms have been developed, there is still no satisfactory solution to this problem. The main obstacles are:

(1) The discrete property of digital images, and

(2) the computation burden.

After a critical review of the algorithms that are currently available for the segmentation of lines and curves, a segmentation scheme will be proposed for use in this work. The implementation of the algorithm will be discussed and results from a series of experiments will be presented in the examples of the next chapter.

5.2 Overview of Algorithms for Line and Curve Segmentation

In general, the algorithms of line and curves segmentation can be classified into three categories: (1) detection of dominant points; (2)line fitting; and (3) transformation. But most algorithms will be shown to fall into the first category.

5.2.1 Detection of Dominant Points

There are many factors that influence human judgment of shape similarity and shape complexity (Zusnes,1970). In the 1950's, Attneave proposed that information of the shape of a curve is concentrated at dominant points having high curvature (Attneave,1954). Since then, many algorithms based on this famous observation have been developed.

5.2.1.1 K-curvature thresholding

One approach is k-curvature thresholding (Rosenfeld and Kak,1976). In this approach points at which the slope lies outside the given range are suppressed. Parts of a curve where the k-slope (or curvature) lies within a given range can be extracted by simple "thresholding".

The k-curvature thresholding method requires the choice of two parameters: K, the number of points for averaging, and the curvature threshold. Choosing a large K value can reduce the digitization noise which can be considerable, but it also reduces the angle resolution.

There are two major problems with dominant point detection on digital curves. One is the precise definition of discrete curvature, the other is the determination of the region of support for the computation of the curvature.

Teh and Chin (1989) explained how to use an iterative algorithm to determine the region of support without inputting any parameters.

5.2.1.2 Angle detection

Another approach is to detect the dominant points directly through angle detection schemes (Rosenfeld and Johnston,1973) (Rosenfeld and Weska,1975). The angles are detected at points where there is an abrupt change in (average) slope.

Constructing chords can also be used to detect angles and local features. One way of using chords is to measure the distance between the point on the curve and the constructed chords. The point where this distance is a maximum would be the best choice for locating the angle (Rosenfeld and Kak,1976).

5.2.1.3 Chord/arc ratio

Another way of using chords is to measure the chord/arc ratio. In essence, this method will give a measure of the straightness of a curve. Wu and Rodd (1992) proposed some criteria for checking linearity and circularity of the object outlines based on analysis of a series of chord/arc ratios.

The difficulty with using chords is how to properly select the k value and to construct the chord. (Suppose given a point P_i of the curve, a chord has to be drawn from P_{i-k} to P_{i+k}). This is the crux of the problem in k-curvature thresholding.

5.2.1.4 Piecewise Polygon Approximation

Piecewise polygon approximation features many times in aoproaches to detect dominant points. It is differs from previous methods in that the curve is broken into line segments. Dominant points correspond approximately to the actual or extrapolated intersections of adjacent line segments of the polygon (Ramer,1972)(Dunham,1986).

5.2.1.5 Corner finding

Freeman and Davis devised a corner finding algorithm for chain coded curves (Freeman and Davis,1977).

Anderson and Bezdek (1984) proposed a different approach to that of previous algorithms in that they defined tangential deflection and curvature of discrete curves, which are based on the geometrical and statistical properties associated with the eigenvalue-eigenvector structure of sample covariance matrices.

5.2.2 Line Fitting

5.2.2.1 Minimum Squared Error Line Fitting

The aim of the Minimum Squared Error Line Fitting (MSE) algorithm is to find the straight line such that the sum of the squares of the vertical distances from each point to the line is minimum.

Given a set of points $\{(x_i,y_i), i=1,...,n\}$ in the plane, the aim is to seek two numbers c_0 and c_1, such that the error function

$$\Sigma^n_{i=1}[(c_0 + c_1 x_i) - y_i]^2$$

is a minimum.

5.2.2.2 Eigenvector Line Fitting

Duda and Hart explained this method in (Duda and Hart,1973).

Both methods assume that all the points in a figure were to be represented by a single line. But, the more general problem is that of describing a figure by means of a set of lines. in this case the figure is partitioned into subsets, such that each subset can be reasonably represented by a single line.

The disadvantages associated with the minimum squared error method and the eigenvector method are the computational cost and the stability of the numerical calculations. The minimum squared error method also suffers from the possibility of mis-fitting of nearly vertical lines with horizontal lines.

5.2.3 Transformation

Another idea to simplify the task of image segmentation is to carry out transformation of the image. There are two ways of achieving transformation; one is coordinate transformation and the other is Fourier transformation. The Hough Transformation is best known method of coordinate transformation.

5.2.3.1 Hough Transformation

The Hough transformation is defined as transforming each of the figure points into a straight line in a *parameter space* (Hough,1962). The parameter space is defined by the parametric representation used to describe lines in the picture plane. In this manner, the original problem of finding co-linear points is replaced by a mathematically equivalent problem of finding concurrent lines. Duda and Hart (1972) proposed an improved method in which an angle-radius was used instead of slope-intercept parameters in the parameter space. Since then, many researchers have applied this method or devised extensions of the method to their line detection algorithms. For example, Cheng, Hsu and Chen used modified Hough transform techniques in an algorithm designed for the recognition of handwritten Chinese characters (Cheng, Hsu and Chen, 1989).

The Hough Transform is quite successful when used in the detection of lines, however, the computation required for testing the lines formed by all pairs of points (say n) is approximately proportional to n^2, and may be prohibitive for large values of n. In addition in certain situations, this approach also has the disadvantage of being sensitive to the choice of coordinate axes in the picture plane.

5.2.3.2 Fourier Transform

The intrinsic equation of a closed curve, which specifies the slope of the curve as a function of arc length, is a periodic function with period L (the arc length). This function can therefore be expanded in a Fourier series on the interval [0,L],

coefficients or combinations of these coefficients can be taken as descriptors of the curve's shape.

This method can be applied to digital curves if the curve is regarded as having a piecewise constant slope. Freeman's chain code provides a good polygonal representation of curves. Each link in the chain code corresponds to a straight line segment whose length is 1 or $\sqrt{2}$, and whose slope is a multiple of 45°. For such polygonal curves, the Fourier coefficients can be expressed as discrete sums. Cooley and Tukey (1965), Ahmed and Rao (1975) described the discrete Fourier transform and Fast Fourier Transform (FFT).

The key point in the determination of a digital straight line is to look for the overall periodicity. Fourier transforms proved to be a better way to doing the job of detecting the periodicity of a signal.

The advantage of using the Fourier transform method to detect the strong periodic nature of an input chain code is that nearly all possible combination between successive periods are checked in a time of order N×log(N).

Lee (1981) applied FFT to determine digital straight line chain code in designing his computer vision system.

5.3 Blob Decomposition Scheme

5.3.1 Design Consideration

One of the aims in designing the machine vision system for this project is that of maximizing automatic operation at each processing stage. This aim requires that

the image segmentation schemes should be efficient enough to be completed within a real time limitation, and be simple enough to be implemented without a human interface.

From the above discussion, it can be seen that the disadvantage of most dominant points detection approaches is the need for the determination of the support region for the computation of the curvature. This still remains as a difficult problem since it is object dependent. Some researches have used iterative algorithms to aid resolving this problem, however, the use of iterative algorithms incurs a considerable penalty in computer time.

The Hough Transform and its extensions usually incur a big computation burden and the co-linear points are found without regard to contiguity. The Fourier Transform method demands the chain code of the object boundary is preprocessed to satisfy the properties in Freeman's conjecture. This technique uses complex-valued functions, which are not a good choice for simplifying the segmentation operations. For a computer to implement a Fourier Transform, global operations on the image are required, in which a new pixel value is obtained as a function of all the pixels of an image. This again requires more execution time. As line processing has been selected as the segmentation method for the system being developed, the use of transformation techniques such as Hough or Fourier will not be required.

In the machine vision system being developed, a blob decomposition approach has been devised based on a basic primitive structure verification scheme. This

scheme breaks up the boundary of the blob to be identified into primitives and verifies the structure type of each primitive by a sequence of operations.. The line verification method being used can be regarded as a modified Minimum Squared Error Line Fitting method.

The primitive structure verification operations are carried out on the boundary of the blob.

5.3.2 The Basic Primitive Structure Verification Scheme

The first step of the basic primitive structure verification scheme is to locate the initial *arbitrary corner points*. The initial *arbitrary corner points* are defined as the intersection points between the blob and the bounding box imposed on the blob. In the case of polygons, the initial *arbitrary corner points* obtained in this way are usually the real corner points. This quite simplifies the corner detection task since in the application being considered, the domain involves only regular geometric shapes. For this reason verification is a relatively simple procedure for automatic implementation

The second step is to carry out structure verification on each primitive. The basic primitives are classified into straight lines, circular arcs, small segments and complex curves. In the system being developed, the classification is constrained within the last three structure types.

If the primitive falls into none of the predefined structure types, the machine breaks up this primitive into two further subprimitives by setting an arbitrary

corner point at the point with largest vertical distance to the primitive that has been verified.

In this way, the blob is described as a chained primitive connected by arbitrary corner points. To establish the real corner points, merging operations are needed. These are operations which merge lines with the same slope, arcs with identical radius and centre, and small segments with an adjacent line or arc.

5.4 Implementation

The blob decomposition algorithm consists of six basic routines: Find_corner_point, Circle_detection, Line_detection, Arc_detection, line_merge, and arc_merge. All the operations of primitive structure verification are carried out on the outline of the blob.

5.4.1 Circle Detection Routine

The input to this routine is an outline of the given blob. The scheme of this routine is to construct a visual circle and then to verify if the outline is identical with the visual circle.

5.4.1.1 Construct a visual circle

The assumed circle takes the gravity centroid (x_0,y_0) as its center. The initial radius is calculated by averaging the distances between the initial arbitrary points

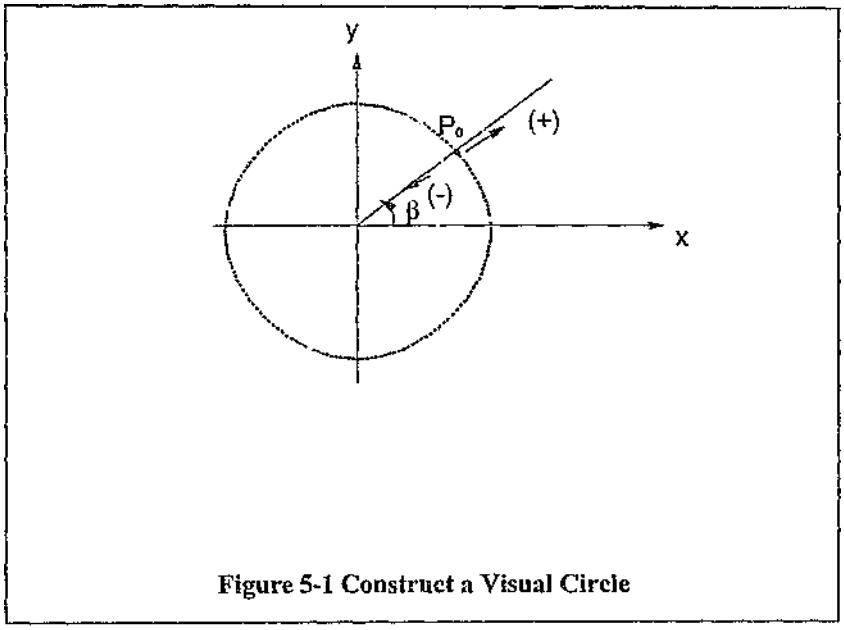


Figure 5-1 Construct a Visual Circle

(P_i) and the center (x_0,y_0).

A pre-filleting out rule is invoked to decide whether the circle detection procedure is applicable. The rule checks to see if the distances between the initial arbitrary points and the center (x_0,y_0) lie within acceptable limits of their average. If not, the algorithm will switch to an alternative primitive verification routine.

5.4.1.2 Verification Procedure

The procedure of the verification scheme is as follows:

(1) Construct a line RR through the centroid (x_0,y_0) the initial slope angle β (see Figure 5-1) is defined as 0° and is increased in 5° steps until β = 360°.

(2) Find the intersection point P(x,y) between the line RR and the blob outline.

To find this point, a searching scheme is devised. Starting at the initial point P_0, which is defined as ersection point between RR and the assumed circle C_0, a search along the line RR is instituted:

$$y-y_0 = x-x_0 + r_0 \tan\beta; \quad (\beta \neq 90°, \neq 270°)$$

step movements in a positive (+) and negative (-) direction are made alternately until a point with pixel value of 255 is identified. When β=90° or 270°, the searching follows a simpler procedure.

This searching scheme is computationally efficient, unnecessary searching is avoided. Once the point P(x,y) is found, the distance between the centroid and this point is computed. When β increases to 360°, a set of distances are produced. The erage of the distances is taken as the r dius R.

ecision Rules:

Rule 5-1: suppose the length of the point on the primitive to th center is dist_r, if the ratio

$$\text{Ratio\_r} = |\text{dist\_r}-R|/R < \delta\_R, \quad \text{(where } \delta\_R \text{ is the given threshold value on the ratio)}$$

then P(x,y) is a point on the assumed circle, otherwise it is regarded as not being on the assumed circle.

Rule 5-2: if the number of points that are considered as being not on' the assumed circle, exceeds a given Number-threshold, the blob is rejected as a circle. Otherwise it's accepted as a circle with center (x_0,y_0) and radius R.

5.4.2 Line Detection Routines

5.4.2.1 Problem Definition

Given $P_1(x_1,y_1)$ and $P_2(x_2,y_2)$ are two endpoints of the primitive P_1P_2 , verify if the primitive P_1P_2 is a straight line.

5.4.2.2 Design Consideration

The method of line verification that was initially used was based on checking the slope ratio $\Delta y/\Delta x$. This routine is defined as follows:

Suppose the co-ordinates of the origin is chosen at point P_1, then if the primitive is a straight line, equation should be of the form y=kx. Thus, if we take the ratio y/x (where y and x are measured relative to P_1) for all points on the line P_1P_2, and if this ratio is approximately constant, it can be concluded that P_1P_2 is approximately a line segment.

However, in practice the experimental results were unsatisfactory for a number of reasons:

(a) The ratio y/x was not a constant, but were in a linear increasing trend.

(b) Near to the origin, P_1 , large errors occurred in the y/x ratio. This situation rapidly worsened as the origin was approached.

(c) The position of origin affected the results.

These phenomena were verified by checking straight lines drawn directly on the image buffer using the computer graphics functions. The ratio is approximately constant and stable only when the slope of the straight line is approximately 45°.

The reason for these errors can be explained as a computer digital error. Referring to Figure 5-2, it can seen that the curve or boundary in the computer representation is approximated as polygons composed of short line segments.

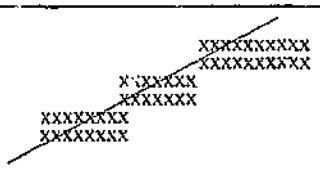


Figure 5-2 Digitized Line

To overcome the difficulties an approach using a modified minimum square error line fitting method was employed. In this approach, only the vertical distance between the points on the blob outline and the chord were checked, instead of using the square computation procedure. In this way the computation burden was reduced and the accuracy of the routine depends on the searching scheme for test points.

5.4.2.3 Line Verification Procedure

The verification approach is to verify if the points on the primitive have acceptable vertical distances to the chord. The procedure can be described as follows:

(1) Assume the straight line is the chord P_1P_2 , its slope is $\tan\alpha = (y2-y1)/(x2-x1)$ ($\alpha \neq 90°$). At P_1 construct a vertical line V. Move the vertical line V in incremental steps Δy, until point P_2 is reached. (See Figure 5-3). The intersection point between P_1P_2 and V is denoted as P_0.

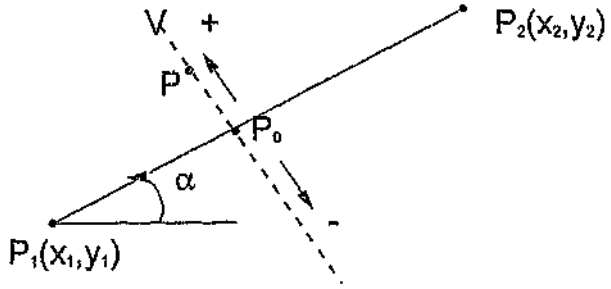


Figure 5-3 Line Verification Scheme

(2) Carry out a search to determine the position of point P on the primitive, i.e., the intersection point between V and the primitive. This is implemented using the following searching scheme:

Starting at the initial point P_0 , the search is carried out along the vertical line V. From this point a shift of one pixel, alternately towards the positive(+) and negative (-) directions is made until a pixel with a value of 255 is found. Selecting P_0 as the initial point gives the most efficient search methodology.

(3) The vertical distance between the point Pa and the chord P_1P_2 is easy to compute once the point Pa is found:

Suppose $P_1(x_1,y_1)$ is a point on the chord P_1P_2, line V is passing through point P_1 (x_1,y_1) and vertical to the chord P_1P_2. Point $P_a(x_a,y_a)$ is the intersection of line V and the primitive P_1P_2 (curve). The angle of chord P_1P_2 is α. The relationship between P_a and P_1 is illustrated in Figure 5-4.

From the right-triangle P_1P_aD, the distance between P_a and the chord P_1P_2 is

$$h = |x_a - x_1| / \cos(\alpha);$$

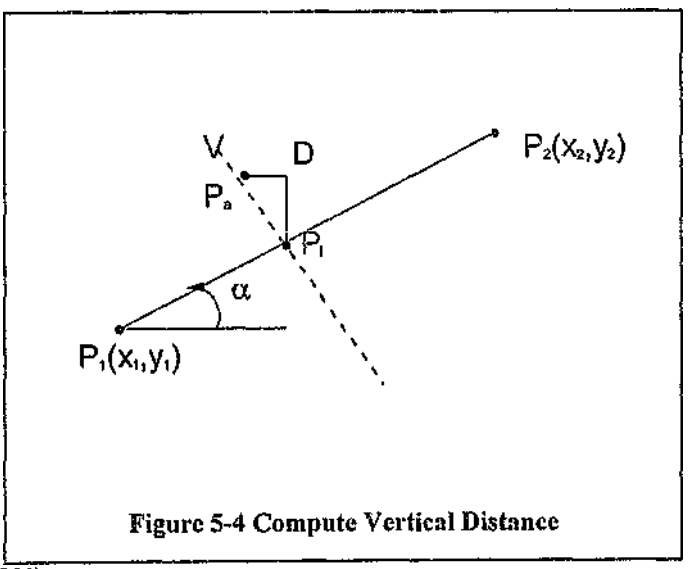


Figure 5-4 Compute Vertical Distance

(where $\alpha \neq 90°$)

In the case where $\alpha=90°$ or within a given domain of 90°, i.e. $[90°-\delta, 90°+\delta]$, the vertical distance can be simply by computed counting the steps it shifts.

The advantage of deriving P_n from P_1 is that there is no accumulated error from other points. Some algorithms use iterative routines to compute parameters, which can cause large accumulated errors.

(4) Decision Rule:

Rule 5-3: suppose the length of P_1P_2 is dd, the vertical distance between point P_n and P_1P_2 is h, if

$$ratio\_line = h/dd < \delta\_L, \qquad (\text{where } \delta\_L \text{ is a given threshold}).$$

then the point P_n is regarded as being on the line P_1P_2. Otherwise, it is regarded as being outside the line.

Rule 2: if the number of points outside the line P_1P_2. exceeds a given threshold value Num_threshold, then the primitive is rejected as a straight line, otherwise, it is accepted as a straight line.

5.4.2.4 Dealing with Blank points and abnormal points

The searching scheme sometimes fails when blank points or abnormal points are produced. This failure also results from image digitization errors. See Figure 5-5. When the vertical line occurs at a discontinuity, either a blank point (a point without value) or an abnormal value is produced.

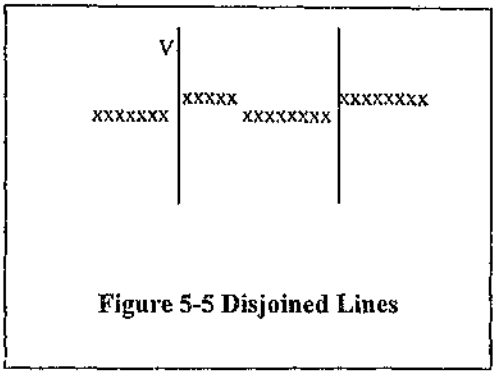


Figure 5-5 Disjoined Lines

A solution to this problem can be obtained by considering the values of adjacent points. Abnormal points can be checked by testing the values of the adjacent pixels, if values on both sides of the point are normal, then this point is regarded as an abnormal point and the value of the adjacent pixel is substituted for the abnormal value.

The blank points can be processed using the values of their adjacent pixels. However, if the number of the blank points is significant then this implies that the image processing operation is in error and the operation has to be repeated in such cases.

5.4.2.5 Check Symmetry

When the line verification routine rejects the primitive as a straight line, the data is utilised for an alternative processing operation as follows. The location of the highest point to the chord is calculated and is used as an indication of symmetry. If the location is found to be at the mid-point of the chord, it is necessary to

invoke the arc verification operation. If this is not the case other strategies have to be considered.

5.4.3 Arc Detection Routine

5.4.3.1 Problem Definition

Given a primitive P_1P_2 , $P_1(x_1 , y_1)$ and $P_2(x_2 , y_2)$, are the two endpoints of the primitive, verify that P_1P_2 is a circular arc.

5.4.3.2 Verification Scheme

The arc verification scheme first constructs a visual circle C_0 and then verifies if the primitive P_1P_2 is part of the circle.

(1) Construct the visual circle C_0 : This is implemented according to the principle that three points can define a circle.

Taking P_1 , P_2 and another point from the primitive will form a circle C^0 assuming the primitive is an circular arc. The simplest way of constructing the circle is selecting the middle point of the primitive as the third point. Finding the middle point of the primitive is accomplished by the following searching scheme:

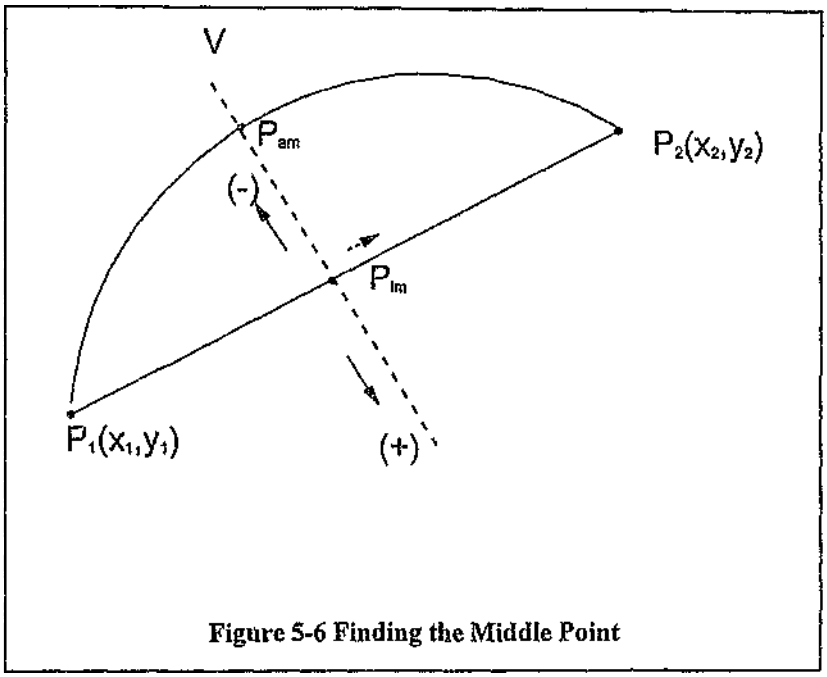


Figure 5-6 Finding the Middle Point

Make a vertical line V, passing through the middle point P_{lm} (x_{lm} , y_{lm}) of the chord P_1P_2 and vertical to the chord, see Figure 5-6. Start searching at P_{lm} (x_{lm} , y_{lm}), and shift alternatively in the positive (+) and negative (-) directions along the vertical line V, until a pixel with value 255 is found. If the search is unsuccessful, due to the search coinciding with a blank point, the vertical line is shifted one pixel to the right.

Experimental verification of the routine described above, however, showed that an assumed circle constructed in this manner yielded large discrepancies from the true value. In order to minimize the error an alternative method where additional circles are constructed, and the results averaged to obtain the centre and radius values was used.

In order to reduce the computation burden, only one other circle, C′, was constructed. To construct the circle C′, a line $P_1'P_2$ is constructed parallel to the chord $P_1'P_2'$ at the mid-point between P_{lm} and P_{am}. In this way, the two intersection points P_1' and P_2' are produced. Using P_1' , $P_2'$and P_{am}, the circle C′ is constructed as before.(see Figure 5-7)

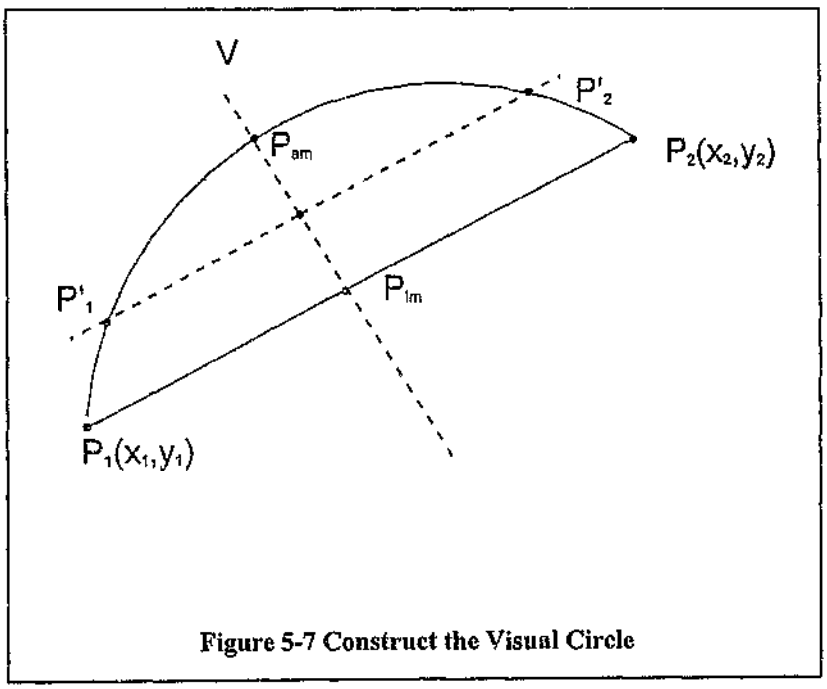


Figure 5-7 Construct the Visual Circle

Suppose the C′ has center (x_c',y_c') and radius R′, the circle constructed through P_1 , P_2 and P_{am} is C^0 with (x_c^0,y_c^0) and R^0 , then the visual circle has center $O(x_c,y_c)$ and radius R, these values are calculated as follows:

$$x_c = (x_c' + x_c^0) /2,$$

$$y_c = (y_c' + y_c^0) /2,$$

$R = (R' + R^0)/2.$

(2) Verification procedure:

Suppose angle P_1OX is β_1 and angle P_2OX is β_2 . Divide the arc angle β_1 - β_2 into n fractions. Make a line RR passing through the center O, with an angle β_1. Turn the line towards the β_2, by starting at β_2 and increasing β with steps of $(\beta_1-\beta_2 /n)$. (See Figure 5-8).

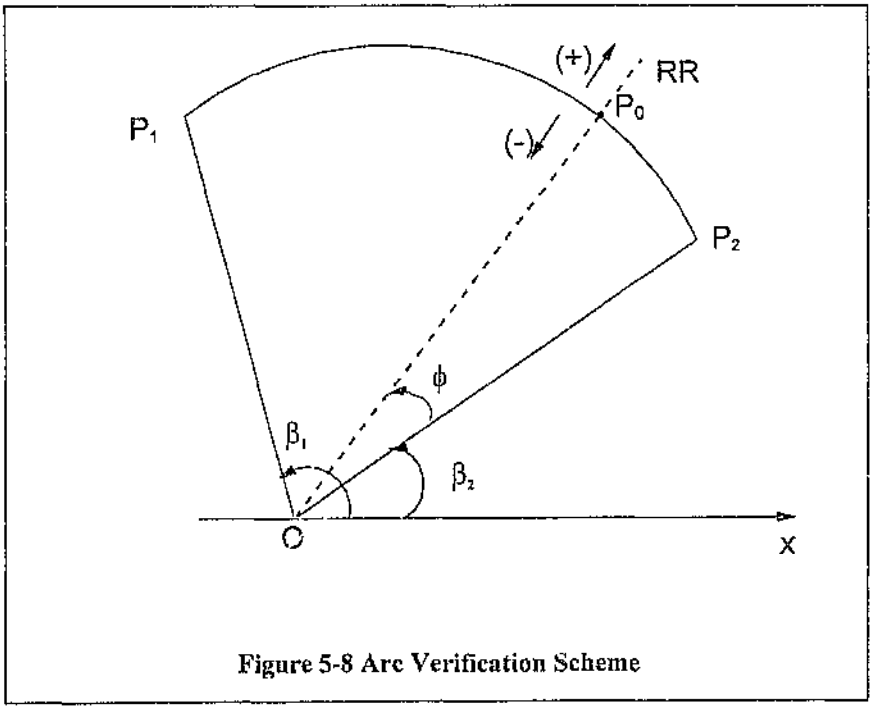


Figure 5-8 Arc Verification Scheme

Find the intersection point between RR and the primitive using the following searching scheme:

Starting at the initial point P_0 , the search is carried out along the line RR. From this point a shift of one pixel, alternately towards the positive(+) and negative (-) directions is made until a pixel with a value of 255 is found.

(3) Decision Rules:

Rule 5-5: suppose the point P on the primitive has the distance rr to the visual center O, if the ratio

$$|R-rr|/R < \delta\_r, \qquad (\text{where } \delta\_r \text{ is a given threshold value})$$

then the point P is regarded being on the assumed circle C, otherwise, it is rejected as a circle point.

Rule 5-6: if the number of points on the primitive exceed a given threshold value, it is not regarded as a circular arc. Otherwise, it is verified as part of the assumed circle C with center $O(x_c, y_c)$ and radius R.

5.4.4 Merging Routines

After the processing using the above verification operation, the blob is described as a primitive chain. The connection nodes between the primitives are called *arbitrary corner points*. Not all the *arbitrary corner points* are real dominant points, as some of these points are simply taken from the bounding box. To establish real dominant points, r. erging operations are needed. The system classes the situations which require merging operations into four categories:

5.4.4.1 Merging Arcs

The bounding box of the blob may produce an arbitrary point on an arc and break

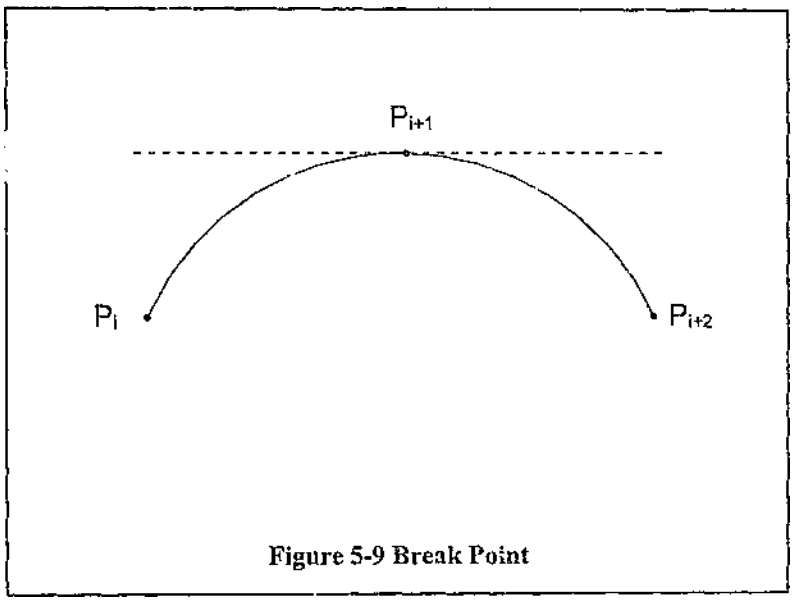


Figure 5-9 Break Point

it into two or more primitives (see Figure 5-9). To deal with such a situation, the algorithm will check through any connected arcs, by discarding the arbitrary point in the middle of the arc and conducting an arc verification operation on the new primitive. If the verification result is positive, the sequence of the arbitrary corner points will be adjusted. If negative, the algorithm will carry on checking subsequent connected arcs until the primitive chain is exhausted.

5.4.4.2 Merging Lines

Due to the image digitization errors, a line can sometimes be represented by a 'saw-like' discontinuity as shown in Figure 5-10. Under these circumstances, a straight line is broken up into several segments. In order to merge these adjacent segments, the Minimum square error line fitting algorithm needs to compare the

slope of ı ɪgment with those of adjacent segments. If adjacent segments are found to have the same, or approximately the same slopes, then a further line verification procedure will be conducted on the segments that have been identified. The segments will be joined form a single line if the verification result is positive. However, the slopes of the small segments are subject to large errors. This problem is dealt with using a special routine for merging small segments which is considered next.

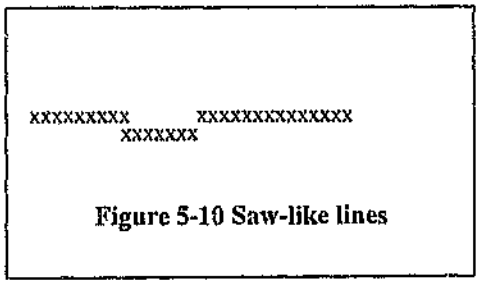


Figure 5-10 Saw-like lines

5.4.4.3 Merging Small Segments

Small segments are often subject to large errors in both the verification operation and in the slope calculation. These situations usually occur as a result of broken lines (Figure 5-10) or as a result of the boundary corner (see Figure 5-11).

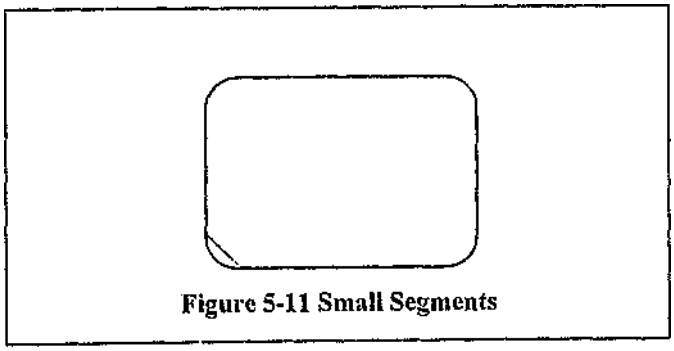


Figure 5-11 Small Segments

In order to form an opinion on the status of a small segment, a decision rule has to be defined as follows:

Rule 5-7: Given that the primitive P_1P_2 has the endpoints $P_1(x_1, y_1)$and $P_2 (x_2, y_2)$, and if the absolute differences Δx and Δy, are within the limits stated as follows:

$$\Delta x = |x_1 - x_2| < \delta\_sm,$$

and $$\Delta y = |y_1 - y_2| < \delta\_sm,$$ (where δ_sm is a given threshold value)

then the primitive P_1P_2 is regarded as a small segment.

Under these conditions the algorithm will merge the small segment with its two adjacent primitives separately and choose the final merge with the one with the lowest error .

5.4.4.4 Merging using feedback information

Small arcs and lines tend to have big verification errors. Such situations will produce contradictory or abnormal results at the system recognition stage. In the system being developed, the Inference Engine has been furnished with an ability to feedback information to the image processing module and also to guide the merging operations (This topic is further dealt with in Section 6.4.3)

The aim of the technique is to merge the smallest arc or smallest line with its adjacent primitives. The smallest arc is defined as the arc with shortest chord length, and the smallest line is the one with the shortest length.

The merging scheme is similar to the small segments merging operation. The ultimate decision for the merging operation is based on achieving the lowest error.

5.5 Blob Feature Extraction

As stated at the beginning of the chapter, this chapter deals with the shape description problem at the blob level. All the discussions before this section have been concerned with the design and implementation of blob decomposition techniques. Blob decomposition is the means of extracting features but, it is not the sole purpose of a machine vision system. The purpose of any image processing operations is to generate a good description for the machine to analyze and understand what it sees.

This section discusses how to use the blob decomposition results to form this description. The result of the blob decomposition can be described as a series of **chained primitives**. The connections between the primitives are a series of **link nodes**, which are regarded as dominant points. The blob description will consist of two parts: one part is the primitive properties and the other is the primitive relationship.

The **primitive properties** are the lengths between two nodes, the structure type, the angle, and the symmetry. A data structure is defined in a program called **Primitive** which implements this description as follows:

```
typedef struct {

        coor_t P1, P2 ;
```

```
    int     type;
    int     sym;
    double bta, len;
}Primitive;
```

The primitive relationship features are the angle between the primitives, the number of lines and arcs, the equality between the primitives such as angles and lengths, along with the geometric features on the blobs obtained from last chapter. These features form a description of a blob. A data structure, **Gen_table** is constructed as follows to accommodate these values in the program:

```
typedef struct {
    int,    no;
    char    *name;
    int     circle
    int     line_num;
    int     arc_num;
    int     eql_num;
    int     eqa_num;
    int     eq_90;
    int     gt_90;
```

```
        int     les_90;
        float   compct;
        float   thin;
        int     area;
        int     peri;
        int     xo,yo;
    }
```

Following calculations on these features, a decision rule is required to establish and define the equality of lines and angles:

Decision Rule 5-8: suppose L_1 and L_2 are the lengths of two lines. $\alpha 1$ and $\alpha 2$ are two angles, if the absolute difference ratio:

$|L_1 - L_2| / L_1 < \delta\_L$,

$|\alpha 1 - \alpha 2| < \delta\_\alpha$, (where δ_L and δ_α are two given threshold values)

then

$L_1 = L_2$,

$\alpha 1 = \alpha 2$.

5.6 Error Analysis

5.6.1 Origins of Errors

The errors of the primitive structure verification routines originate from two sources. One is the discrete property of the digital image, the other is as a result of the edge segmentation operation.

The 'saw-like' approximation can present serious problems when segments are oriented at an angle in the region of 45°, in this situation the errors in the line verification algorithm tend to assume their largest values. (See Figure 5-12)

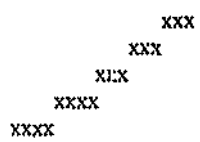


Figure 5-12 Line with a big Approximation Error

For the circle and arc verification operations, the precision of constructing visual circles is crucial, especially for the arc verification scheme. When the arc is small, the error can be critical.(See Figure 5-13).

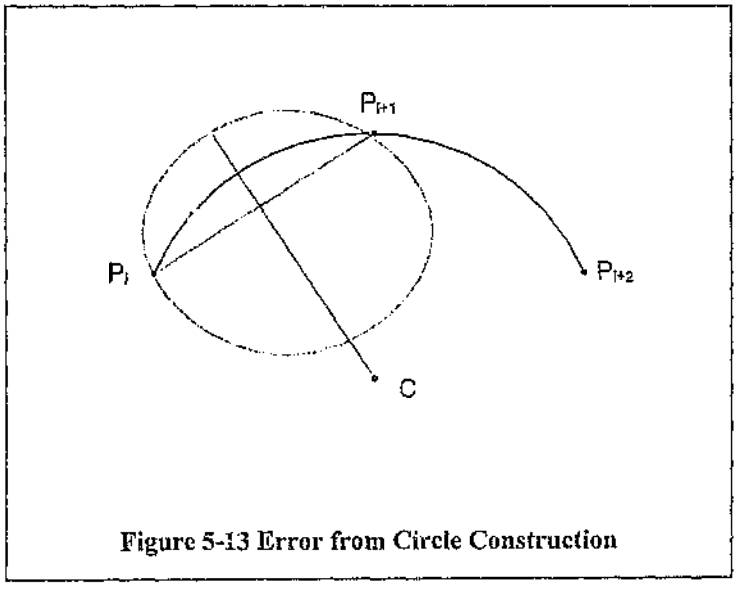


Figure 5-13 Error from Circle Construction

5.6.2 Set Error Thresholds

An allowance has to be set to accommodate some degree of error. For this purpose threshold values have been set in the program. In the decision rules, there are normally two kinds of threshold values; one is for the difference ratio , the other is a number allowance, in certain cases this is defined as a number difference ratio.

The advantage of using a ratio instead of a practical distance or angle value, is that it is insensitive to the scale of the shapes being considered.

The error thresholds act as filters. Too large a threshold value may make it difficult, if not impossible, to distinguish the full range of features, while too small an error threshold may lead to the verification operation breaking up the scene into too many subprimitives and complicate the merging operations. . Setting appropriate error threshold values is based on a trial and error procedure based on experimentation.

In experimental work carried out on this project, it was established that the system performed efficiently when the length difference ratio was set at a value between 0.025 and 0.05, the angle ratio at 0.1, and the number allowance at either 3 or 4.

5.7 Discussion and Conclusion

The low level image decomposition uses the blob bounding box to find the initial arbitrary dominant points and then carries out a series of basic primitive structure verification operations on the blob boundary. The blob to be verified is extracted from the image as a chained primitive.

The basic structure verification approach is a modified minimum squared error approximation method.

This method has similarities to the chord/arc ratio method used by Wu and Rodd (1992), however, the verification scheme used in this project differs fundamentally from that work. In the work presented here the chord is used to aid the construction of visual circles instead of building the chord/arc ratios as checking criteria.

If the dominant points and angle detection methods are compared, the structure verification approach proposed here is simpler to implement and has a high correlation ratio within the application domain being considered.

6. BUILDING THE KNOWLEDGE BASED SYSTEM

In this chapter, the details of the design and implementation of the knowledge base section of our machine vision systems are explained. The chapter begins with an introduction to the basic structure of the knowledge based system and its application to machine vision systems. Then, the forms and approaches in which image shapes are expressed in the knowledge base and how to manage the knowledge base are discussed. Design considerations and strategies behind the construction of the Inference Engine are given to show the benefits that are obtained using this design methodology. In the last section a selection of the experimental results are presented together with deductions and some conclusions based on those results.

6.1 INTRODUCTION

In this section, following an overview of the knowledge based system - its basic structure, its development history and applications, the reason why the knowledge based system technique is chosen to solve the machine vision problem is explained. References to previous work which used this technique are given as evidence of its utility. Finally, the architecture of knowledge base in the system will be described.

6.1.1 An Overview of Knowledge-based System

A knowledge-based system, popularly referred to as an expert system, is a computer system that exploits the specialized knowledge of human experts to achieve high performance in a specific problem area, or domain.

A block diagram of a typical knowledge-based system is shown in Figure 6-1. The internal structure of the system consists of three major regions: a Knowledge Base, a Data Base, and an Inference Engine.

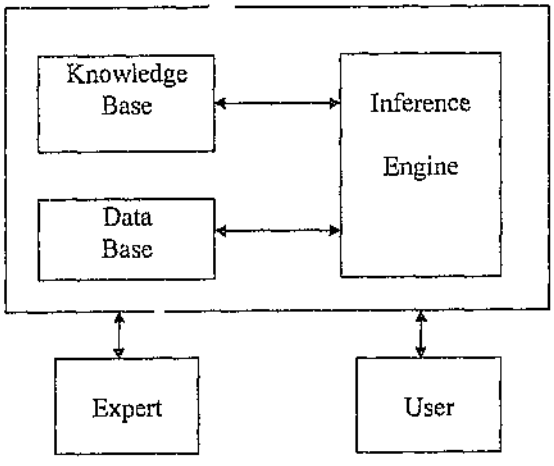


Figure 6-1 Basic Structure of Knowledge Based System

The Knowledge Base contains the rules of inference that are used during the reasoning process. The Data Base contains the facts available to the system at any given time.

The Inference Engine gui easoning process through the knowledge base by attempting to match the facts in the data base to the rule conditions. In addition to controlling the matching process, the inference engine provides conflict resolution when more than one production rule is matched with a given set of conditional elements in the data base.

Specialized knowledge can be acquired though experts by a man-machine interaction.

Though the knowledge-based system is a modern technology, its design concepts are as ancient as civilization. As early as 700 BC, the Babylonian scholars developed an elaborate set of "If-Then" rules describing the empirical associations between observable phenomena in everyday life. Many of these constituted an early attempt to develop a system of medical diagnosis.

The idea of a mechanical device to make human like judgments and process information, is an old dream of human beings, predating the advent of electronic computers. During the World War II, Alan Turing proposed the now famous "Turing Test" for machine intelligence, in which a machine imitates a human in a question-answering session. The topic of Turing's paper is whether machines can think, and how we might try to provide an objective answer to that question (Turing,1963).

In 1945, Vannevar Bush(1945) described a device " in which an individual stores his books, records, and communications, and which is mechanized so that it may

be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory..."

However, Turing's speculations, and work by other researchers, were shelved due to the more pressing needs of wartime applications.

The early 1950s marked the renaissance of Artificial Intelligence, this was partly promoted by the development of hardware technology. Research teams in both academic and commercial environments began exploring this new field.

In 1958, McCarthy (1958) offered a fairly concrete proposal for a program, Advice Taker, that accepted new information with which it reasoned about its actions, thereby improving its performance. It was the first time that a technical framework had been provided for the construction of systems that built up a knowledge base incrementally and that reasoned with respect to that knowledge base for the purpose of performing certain tasks. However until the 1960s, the idea of a machine that could reason remained a theoretical concept.

In 1971, AML International, a company specializing in medical laboratory automation, introduced the first computer-based medical diagnostic system. The AML project not only contained a broad base of medical expertise, but also interfaced directly with diagnostic instrumentation, such as spirometry, blood pressure, and optometric equipment.

In 1974, the most famous precursor to the modern rule-based knowledge base system emerged from the Stanford Heuristic Programming Project (HPP). The core of the project was an exhaustive knowledge base, a simple rule-based

processing method, and a symbolic math. It was a working model for the virtually all expert systems in use today.

By far, knowledge based systems are the largest application area in the field of artificial intelligence. The applications range from diagnosing disease, to evaluating mineral deposits, analyzing chemical compounds, and troubleshooting circuits, to military, academic advisement, tax advisement, computer-aided design and computer-aided manufacturing, VLSI design, office automation, and engineering. Knowledge based systems have found applications in almost every field that requires specialized knowledge.

6.1.2 Design Motivation

How do we judge a person's intelligence? The question is usually in terms of a persons capacity for knowledge and his ability to use this knowledge to solve problems. The aim of this project is to develop an intelligent machine vision system. Two types of knowledge are required for this purpose:

(1) The knowledge about what is seen, and

(2) The knowledge of how to understand what is seen.

The first type of knowledge can be obtained by the techniques described in the Chapters 4 and chapter 5. In machine vision this is the area of image processing.

The second type of knowledge is called Image Understanding, or Machine Perception, is by far the most complicated of all machine vision tasks.

6 BUILDING THE KNOWLEDGE BASED SYSTEM

Even though the imaging devices and image processing technology has made great advances, in some degree excelling those of human visual organs, machine perception is far below the abilities of human visual perception.

This is the reason which has led to the development of, and studies in, Psychophysics. One theory of these studies popularly accepted by researchers, is that humans think by recognizing patterns, in short term memory. Such image information triggers a long-term memory that carries out a particular thought process. The short-term memory is updated by sensory information and the results of long-term memory processes. As long-term memory is updated, old information is passed to long-term memory or forgotten.

Inspired by the idea of emulating this thought process, we postulate that using a knowledge base, served in a similar manner to human memory, coupled with efficient organization of knowledge and control strategy, the image understanding task for the machine vision system will be enhanced.

There have been some efforts to apply expert system technology to computer vision systems. Gregory(1984) used a 'Knowledge Based Model' to direct the image analysis in an example from automobile industry. The performance can be improved while allowing programming to be performed by non-expert users in a graphical manner. Lemmer and Mitchell (1984) discussed their work on the development of an interactive environment for developing rule bases used to capture knowledge required for feature identification. Neumann(1985) showed by his work that adopting a 'knowledge-based' approach can lead to design systems

with extended applicability and predictable performance. He also discussed that the process of adapting a vision system to a new task can be facilitated by 'configuration experts', i. e. expert systems guiding the configuration process. Agapakis and Masubuchi (1985) proposed uses of machine vision for the determination of the weld joint and bead geometry, and developed a first prototype of a rule-based expert system for the interpretation of the visually detected weld features and defects. Dixon(1986) used an Expert System layered on top of his system to guide the naive user through the process of generating a vision application. Duane (1986) developed a rule-based expert system to segment and label the major elements in digitized images of simple road scenes. The system acts on a set of uniform regions, generated by the split-and-merge algorithm, which may be regarded as a non-standard primal sketch. Simple descriptors attached to regions and to boundaries between regions are referenced by the rule base, which incorporates general knowledge of the typical behavior of the split-and-merge algorithm, as well as specific knowledge of the scene domain. The rules merge oversegmented regions to form an essentially correct description of the scene. Wesley (1986) discussed some work on integrating the DS theory into a knowledge-based high-level computer vision system. Solinsky (1986) developed a model-based optical character recognition system. The decisions are based on the knowledge incorporated into an 'expert' system set of logical production rules. Dutta Majumder(1988) presented an overview of the framework of current image understanding research from the points of view of knowledge level, information level and complexity by giving examples of industrial vision

system and scene analysis in aerial photography. Dengel (1989) presented a knowledge-based document-analysis system. The system uses a formalism for document layout description and modeling. The model is realized by a tree structure, which describes the layout of a document page in different layout abstraction levels.

Nazif and Levine (1984) described a rule-based segmentation system to create effective segmentation algorithms. Production rules, regarding the splitting and merging of regions and lines, served as the basis for the inference mechanism.

Radig (1992) proposed a knowledge-based approach for recognizing objects in a traffic scene by generating a sequence of vision routines.

6.1.3 Architecture of the System

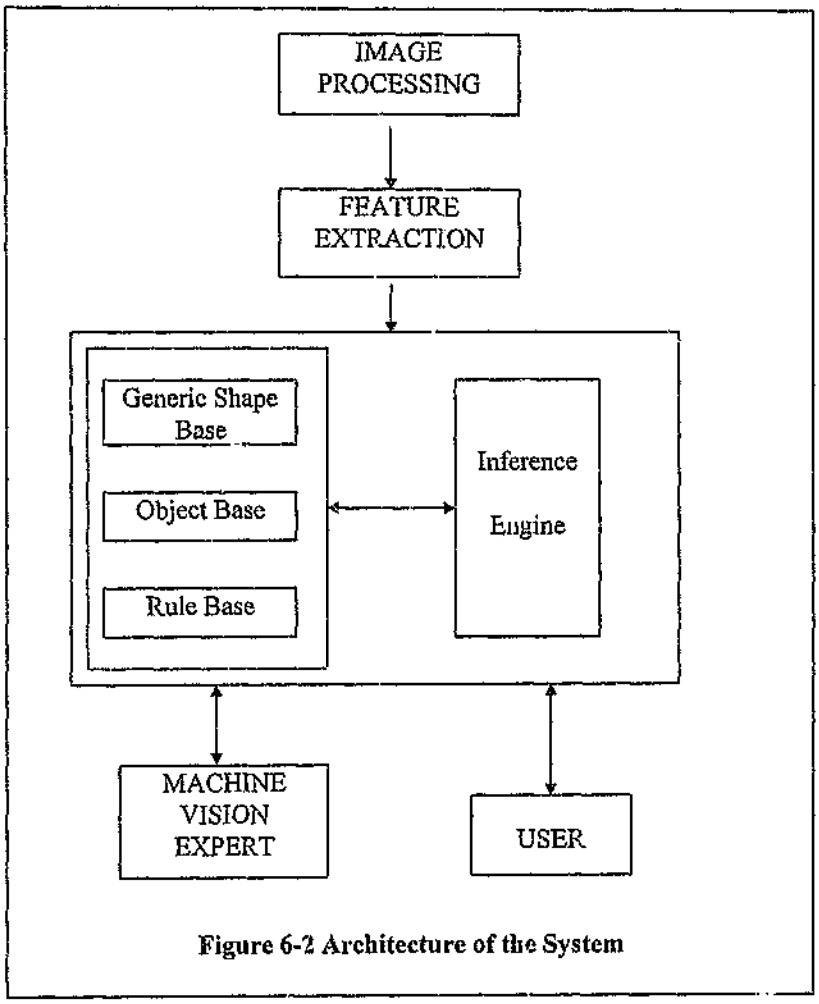


Figure 6-2 Architecture of the System

In the work reported here an intelligent machine vision system was implemented by applying knowledge based technology. Figure 6-2 shows the architecture of the knowledge base section of the system. The knowledge base consists of three bases: Generic Shape Base, the Object Base, and a Rule Base.

The Generic Shape Base and the Object Base are built up using the knowledge of a machine vision expert as static data. The Rule Base includes the inference logic and control rules. The Inference Engine is a mechanism to guide the operation sequences of the whole system.

Building the knowledge-based system involves two steps:

(1) Acquiring the vision knowledge and representing it in a form acceptable to the computer.

(2) Developing the Inference Engine.

In the following sections, a brief review is given on knowledge representation techniques. Based on this review, the shape representation method in the knowledge base of this project will be explained.

6.2 Expressing Shapes in the Knowledge Base

This section discusses the problem of how to express shapes properly in the knowledge base. Based on the achievements of Knowledge Representation Engineering in AI, and a brief review on general object description techniques, a shape expression scheme is proposed, which is a combination of some of the knowledge representation techniques.

6.2.1 An overview of Knowledge Representation Techniques

Two key elements of Artificial Intelligence are the knowledge and reasoning using this knowledge. However, it is useless if the knowledge can not be processed by

the computer. For this reason, knowledge representation has been one the most active areas of Artificial Intelligence research and development.

Knowledge representation techniques can be roughly categorized into four classes: Logic, Production System, Semantic Networks, and Frames. Some of the other methods used to represent knowledge can be reduced to one of these four major techniques.

Logic is a formal knowledge-representation technique. Knowledge is represented in logic using proposition and predicate calculus. Prepositional calculus represents knowledge using statements, called Propositions, which are either true or false. Predicate calculus extends the idea of propositional calculus by using variables, quantifiers, and functions within the logic statement. In logic, inferences are made from known facts to produce new facts that are always guaranteed to be true. Two very powerful laws of logic that allow reasoning are resolution and unification.

A **Production System** is a collection of IF/THEN statements, which are called Rules, or Productions. The IF part of the production represents the conditions, to activate the THEN portion of the production. The condition portion of a production is called the left-hand side, LHS, and the action part of the production is called the right-hand side, RHS.

Figure 6-3 illustrates the principle of a production. A computer searches through the Rule Base and tries to match the production elements against its Data Base. When all the elements of a given production are found, the production is fired. Its conclusion is used to update the computer Data Base.

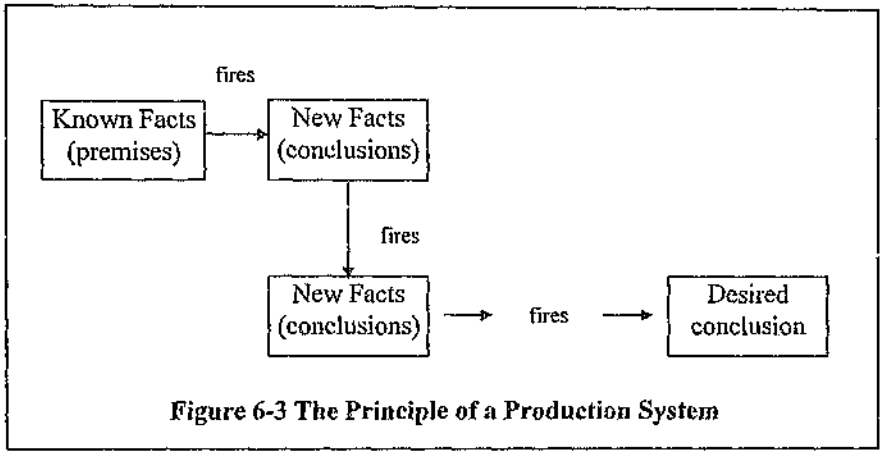


Figure 6-3 The Principle of a Production System

The major reason that production systems are popular, especially in expert systems, is that human knowledge is easily represented using the IF/THEN production rules. Another big advantage of Production Systems is that they easily lend themselves to verification and explanation of the results and conclusions.

As a knowledge representation technique, Production Systems allow knowledge to be added, deleted, or changed without consequence to the rest of the system. This manageable characteristic makes Production Systems most attractive.

The biggest disadvantage of Production Systems is that a comprehensive knowledge base requires a large number of production rules. As a result, problem solving using this technique tends to be inefficient.

A Semantic Network is a network of symbols of that describes the relationship between elements of knowledge. It consists of Nodes and Arcs that link the Nodes. The Nodes contain objects, or situations in the problem domain. The Arcs,

also called Links, represent relationships. Figure 6-4 illustrates a Semantic Network describing a shape.

The inheritance property of Semantic Networks permit deductive reasoning and conserves memory space within a computer system. A computer system reasons by simply following the links. The links act as pointers within the program execution.

A disadvantage of using Semantic Networks for representing Knowledge is that the notation and reasoning process is not formalized. Thus, no common principles can be applied to all networks.

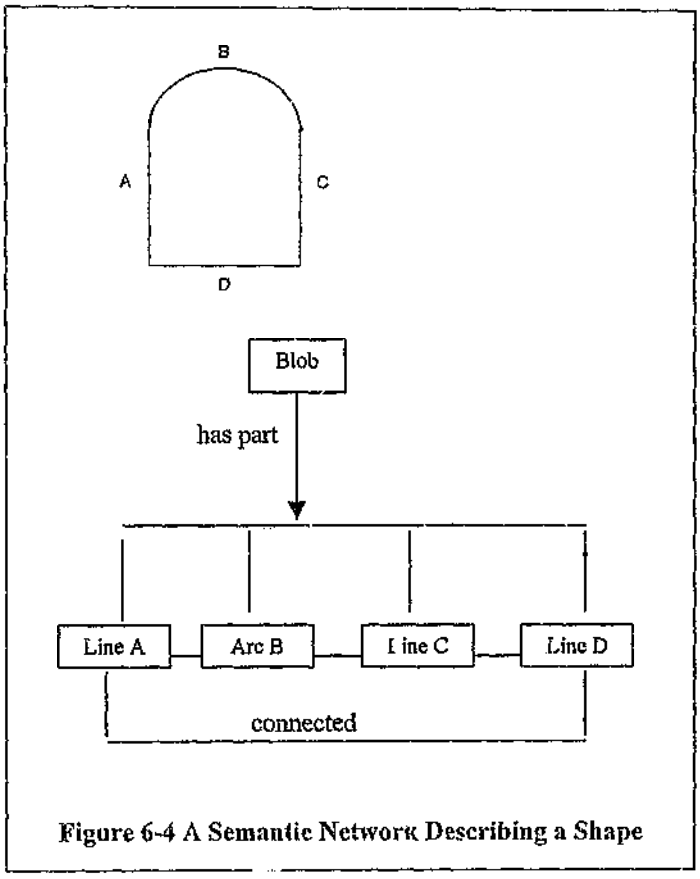


Figure 6-4 A Semantic Network Describing a Shape

Frames theory was proposed by Marvin Minsky of MIT in 1974, as a technique for representing large amounts of general purpose knowledge.

A **Frame** is a data structure that consists of exceptions for a given situation. It contains objects and facts about a situation, as well as procedures on what to do when a given situation is encountered. The knowledge associated with a frame is contained in slots. Reasoning is accomplished by filling the frame slots. Figure 6-5 shows the basic structure of a Frame.

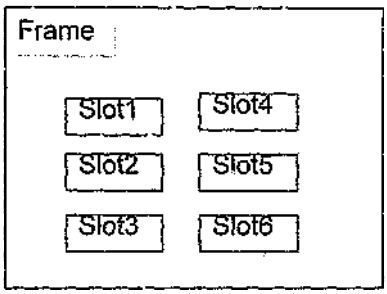


Figure 6-5 Frame Slots

Frames provide a natural means whereby computers can understand their environment and learn from experience, particularly for robot navigation applications. However, frame theory is an abstract idea, and thus requires much more research before it can become a practical means of knowledge representation.

6.2.2 Object Description Methods

The above sections give an introduction to the common methods used to represent general knowledge by machines. In machine vision systems, the major source of knowledge is from an image scene. Therefore, the problem of knowledge representation in machine vision systems is mainly dealt with using object description methods.

The selection or design of a representation scheme depends crucially on the purpose for which it is intended. A certain type of representation may be ideal for one application yet next to useless for another.

Most object description methods fall into two major categories. One scheme is designed to "approximate" the geometry of objects using modeling techniques developed mainly for computer graphics and CAD applications, where the aim is to produce models with sufficient details to support rendering and manufacturing.

The other is the most common way to describe objects in Pattern Recognition, in which object description is also an important issue. It is not aimed at approximating an object but rather makes explicit salient features relevant to a particular task. An object is described through a Feature Vector. Typically, for each object in the object set, n measurements are made for its *n* features, and the result is considered as a point in an n-dimensional feature space. A Feature Vector is formed by joining the feature measurements.

The **feature vector** includes information on feature name, feature value or values, and explicit or implicit information on relationships among features, if these exist. Numerical data and symbolic data are the representation formats for feature vectors. The choice of the object representation format, whether numerical or symbolic (linguistic), is influenced by the pattern recognition methods employed to resolve the classification task.

6.2.3 Selection of the Shape Description Method

Based on the foregoing discussion, it is felt that a combination of some of those knowledge representation methods will be of assistance in solving the shape expression problem for the machine vision system being designed for this project.

The shape expression scheme selected consists of a combination of **the semantic network, feature** vector and production system.

Using the concept of the Semantic Network, objects are segmented into generic blobs. Figure 6-6 illustrated how an image scene is described by this method. Figure 6-6b is a network description of the image in Figure 6-6a.

For each generic blob, a feature vector is employed to represent it. The feature

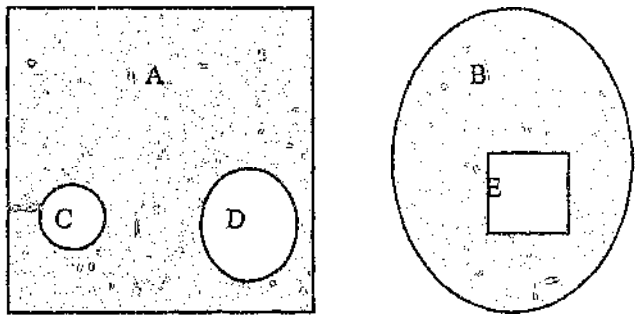


Figure 6-6a

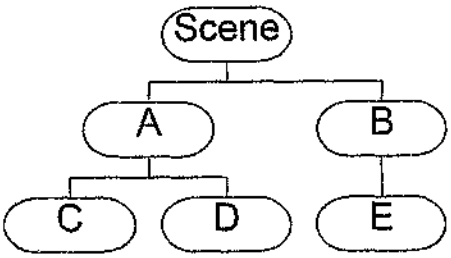


Figure 6-6b

Figure 6-6 Network Description of Image

vector contains both numeric and symbolic information on the blob.

The Production System was chosen as the control mechanism for the sequencing action in the system, because of its characteristics of easy use and of expansion. As limited rules are involved in machine vision systems, the disadvantage of the inefficiency that results from large numbers of production rules can be avoided.

The advantages of this object description method are:

(1) Clarity: objects are described not only by a set of numbers, but also by some meaningful words. The user knows the reason why the system fails and/or succeeds.

(2) Ease of use: both the system designer and user can access the knowledge base, making interfacing with the system possible.

(3) Ease of expansion: new objects and rules can be incorporated into the system, which makes later system expansion convenient.

6.3 Management of the Knowledge Base

The above section details the solution to the problem of optimal shape expression in the knowledge base. This section will discuss the development of a mechanism for feeding facts and rules into the knowledge base, maintaining the repertoire of facts and displaying the contents for periodic review by the human engineers and operators of the system.

6.3.1 Structure of Data Base

The data structure used to represent the objects in the Object Base and the Generic Shape Base are arrays. Each item in the array stores information about the object or the blob. Figure 6-7 shows the structure of the data base.

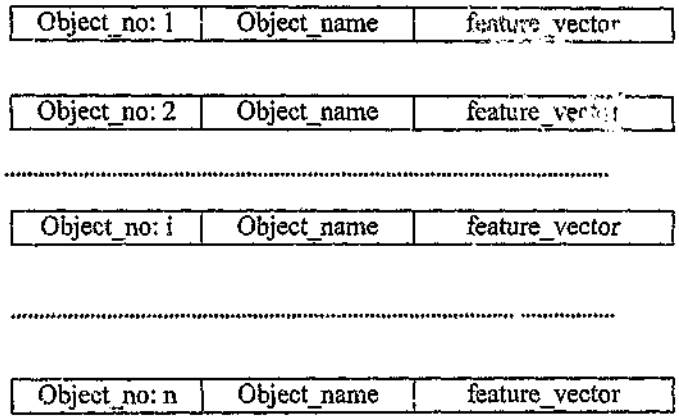


Figure 6-7 Structure of the Data Base

In the Object Base, an object is defined by an Object-Table, which defines a data structure that stores the information on the object's name, features and relationships between its sub-blobs. Table 6-1 is an example of an object table.

Object_no.	
Object_name	
Number of Holes	
blob_shape_no[1] blob_shape_no[2] ..... blob_shape_no[n]	
ratio_area_of_hole[1] ratio_area_of_hole[2] ..... ratio_area_of_hole[n-1]	
.....	

Table 6-1 Example of an object table

For the Generic Shape Base, a data structure **Gen-Table** is defined to store the information on the blob. It contains three fields: Blob number, Blob name, blob dominant features. Table 6-2 shows an example.

Gen_shape_no	
Gen_name	
number_of_lines	
number_of_arcs	
line_with_same_length	
number_of_same_angle	
angle_equal_90°	
angle_big_than_90°	
angle_less_than_90°	
ratio_compactness	
ratio_thinness	
blob_area	
blob_perimeter	
gravity_centriod	
.....	
.....	

Table 6-2 Example of a Gen_Table of the Knowledge Base

6.3.2 Implementation and Maintenance

The knowledge based machine vision system must have a mechanism for inserting facts and rules into the knowledge base, maintaining the repertoire of expressions in the knowledge base, and outputting the contents of the knowledge base for review by the system designer and users.

To accomplish this requirement, a mechanism for adding new objects or new generic blobs to the Object Base and Generic Shape Base respectively is needed. Then, a means of modifying an object description or a blob description and their

values is required. Finally, a method of locating and retrieving a specified object or blob and values for review must be provided.

For the Object Base, four basic modules are constructed to handle these functions:

ADD_OBJECT provides an instrument for adding a new object to the Object Base.

FIND_OBJECT searches the Object Base for a specific object name.

DISPLAY_OBJECT displays all the object names and values in the Object Base.

MODIFY_OBJECT permits the modification of object names and values.

For the Generic Shape Base similar basic modules are provided to implement these functions:

ADD_BLOB provides an instrument for adding a new blob to the Generic Shape Base.

FIND_BLOB searches the Generic Shape Base for a specific blob name.

DISPLAY_BLOB displays all the blob names and values in the Generic Shape Base.

MODIFY_BLOB permits the modification of blob names and values.

6.4 Inference Engine

This section discusses the design and implementation of the Inference Engine in the system. The function of the Inference Engine is to provide a mechanism

capable of performing two basic tasks - facts matching and fact searching. This function is explained in detail, following which the various matching techniques, especially in pattern recognition will be discussed. Based on this discussion, a facts matching approach for our Inference Engine is proposed. The reasoning strategies behind the Inference Engine are also explained with emphasis on the provision of feedback information.

6.4.1 Function of the Inference Engine

The Inference Engine is the rule interpreter that applies the facts in the knowledge base to solve specified problems. It performs this feat simply by formulating trial hypotheses and testing them against a stipulated goal.

Thus the reasoning function of the Inference Engine can be regarded as a process of searching and matching. It directs the computer to search the conditional part of the production through the knowledge base, determines which productions match this conditional part and then decides the order in which the productions are to be fired.

6.4.2 Matching techniques

In a machine vision system, the facts which the inference engine searches for are encoded descriptions of image shapes. Therefore, the matching of facts is not as simple as that carried out by the majority of knowledge based system applications. Correct matching between image objects is the major issue for Pattern Classification.

Pattern Classific tion is a study to classify an object into one of a number of groups or clusters. There are two classical methods in Pattern Classification. One is Statistical Classification. The other is structure classification, also referred to as the linguistic or syntactic approach.

6.4.2.1 Statistical Classification

The basic paradigm of the Statistical Classification is to make n measurements on the object to be classified. The result is a feature vector. Each object is represented by its feature vector and is considered as a point in the n-dimensional feature space. The purpose of the statistical classification is to divide the feature space into compartments and assign the object to be classified to a group corresponding to the compartment into which its feature vector falls (Duda and Hart 1973).

Nearest-Neighbour is the straightforward method to classify an unknown vector **x**. It finds the nearest example and notes the class that it belongs to.

Definition 6.1:

Let the feature vector of the j^{th} sample in the i^{th} class be $\mathbf{x}_{ij}$.

if for some k and l,

$$| x_{k,l} - x_{i,j} | < | x_{i,j} - x | \text{ for all i and j,}$$

then the unknown **x** is assigned to class k.

Nearest-Centroid Classification is a simple way of dividing up the feature space. It supposes that the examples of each pattern class form a nice round cluster and that clusters corresponding to different classes are similar in extent and do not

overlap too much. In such cases, the centre of mass is used to represent each class. An object is then considered to belong to the class whose Center of mass is nearest to it.

Bayes' Rule is the basis of all statistical classification theory. It uses conditional probability to assign an object to the class that it comes from.

Definition 6.2:

If an object may come from any one of m classes C1, C2, ...,C_m , and its feature vector is $\mathbf{x} = (x1,x2,... , x_n)$, then Bayes' Rule says:

Assign to class i where

$$P(C_i \mid \mathbf{x}) > P(Cj \mid \mathbf{x}) \text{ for all } j <> i$$

(that is, assign to the class with the largest conditional probability of membership.)

Assume that the probability of obtaining a given measurement within a class follows a known distribution, the probability of getting a given measurement from class Ci is

P(**x** | Ci) and the following formula is known as **Bayes' law**:

$$P(C_i|x) = \frac{P(x|C_i).P(C_i)}{\sum_j P(x|C_j).P(C_j)}$$

Bayes' law provides the only really practical way of applying Bayes' rule.

Bayes' rule is the best classification rule because it results in the smallest possible average error rate. However, in general, the distribution of the measurements in each class is unknown.

It commonly assumes the distribution for P(x|Ci) follows the normal distribution, because the normal distribution often occurs in practice either exactly or approximately, and it is also a good approximation to a wide range of other distributions.

If, in addition to being normally distributed, each pattern class has the identical covariance matrices, the resulting Bayes' rule takes the form of linear discriminant functions.

Linear classification is by far the most common sort of classification rule used in pattern recognition and classification. However, a linear classifier can only work well with classes that are widely separated or when the classes are elliptically symmetrical or approximately so.

6.4.2.2 Syntactic Pattern Recognition and Classification

Syntactic Pattern Recognition and Classification represents a pattern by the hierarchical (tree-like) structure information of the pattern. That is, a pattern described in terms of simpler subpatterns and each simpler subpattern again can be described in terms of even simpler subpatterns. This structural approach of representing patterns was proposed because of the need to describe complex pictures in the scene analysis. In a complex scene analysis field, the patterns

under consideration are often very complex and large amounts of features are required (Fu, 1976).

The **syntactic approach** can be understood by an analogy between the (hierarchical, tree-like) structure of patterns and the syntax of languages. Patterns are specified as being built up of subpatterns in various ways of composition, just as phrases and sentences are built up by concatenating words and words are built up by concatenating characters.

The simplest subpatterns selected, which are much easier to recognize than the patterns themselves, are called "**pattern primitives**". The "language" which provides the structure description of patterns in terms of a set of pattern primitives and their composition operations, is called "**pattern description language**". The rules governing the composition of primitives into patterns are usually specified by the "**grammar**" of the pattern description language.

The recognition or classification process is accomplished by performing a syntax analysis or parsing of the "sentence" describing the given pattern to determine whether or not it is syntactically (or grammatically) correct with respect to the specified grammar.

Definition 6.3:

A grammar G is a four-tuple

$$G = (V_N, V_T, P, S)$$

where

V_N is a finite set of nonterminals,

V_T is a finite set of terminals,

$S \in V_N$ is the start symbol,

and P is a finite set of rewrite rules or productions denoted by

$\alpha \rightarrow \beta$.

The advantage of the syntactic approach to pattern recognition and classification is that it provides a capability for describing a large set of complex patterns using small sets of simple pattern primitives and of grammatical rules. Besides it makes it possible to express some basic structural characteristics of an infinite set of sentences in a very compact way, since a grammar rule can be applied any number of times.

6.4.2.3 The Approach adopted for this Project

The facts matching approach for the Inference Engine in our machine vision system, consists of two basic verification algorithms; one is **Object Verification,** the other is **Blob Verification.**

The **Object Verification** is a two-phase scheme. Once the Object_Table is obtained from the image analysis module, the algorithm searches the Object Base for a close match. It starts by checking the **dominant features** of the object in the Object Base, and stops at the object with the same dominant features. This is the first phase of the verification.

The **dominant features** are introduced into the matching approach. These features are used to control the search flow in order to avoid unnecessary computation. This idea is borrowed from the concept of the Difference Reduction Search technique in Artificial Intelligence. The difference reduction technique requires that the search always progresses in a direction that minimizes the difference between a given state and the goal state.

Taking object verification as an example, the number of holes in the Object_Table is used as a dominant feature. The searching algorithm only stops at objects consisting of same number of holes. In this way, only the object with closest match is chosen.

In Tables 6-1 and 6-2, the shaded entries are designated as dominant features.

At the second phase, the searching algorithm runs through all blobs connected

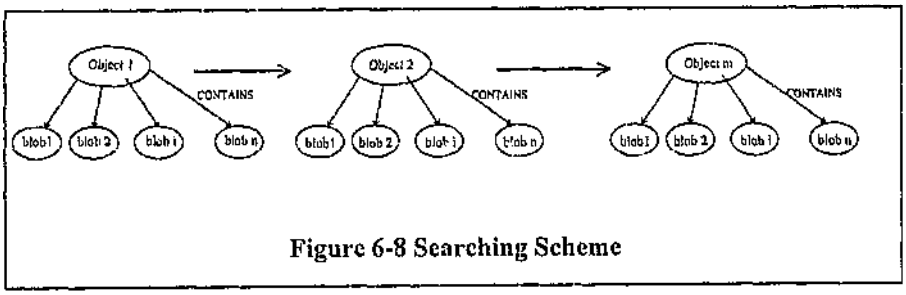


Figure 6-8 Searching Scheme

with the object by the " CONTAINS " relation. Once any blob is found not to match with the given Object_Table, the algorithm will return to first phase searching, and carry on to the next object, until all the objects in the Object Base are exhausted. See Figure 6-8.

Blob Verification is an algorithm to check if a blob with a known Gen_Table matches a generic blob in the Generic Shape Base. It starts at the blobs with identical dominant features. The dominant features used for the blobs are the number of lines and the number of arcs.

Rules are employed in the blob matching method. The advantage of using rules for matching is that rule descriptions are flexible and tolerate errors that have resulted from image processing and feature extraction. For example, when the number of arcs is zero, and the number of lines is 4, the algorithm switches to the tetragonal verification branch. In some cases, even when the number of lines with identical length is 4, the number of angles with 90 might be only 3. This problem can be solved by setting a rule such as:

If the number of lines with identical length is 4,

and the number of angles of $90^{\circ} >= 2$

Then the blob is a square.

Besides the **dominant features**, other features such as **flexible features** and **ratio features** are catered for. Different rules are applied to the comparison of these features.

The **flexible feature** as its name implies is a measurement that does not have a fixed value. It is used as a reference to assist in making shape decisions. Examples of flexible features are gravity centroids, areas, perimeters, etc. The same kind of shape might have different values for these features. However they can be used as an aid to give details on a specific object and its location. Therefore, **flexible**

features are not used to compare differences between the object or blob to be verified and the model in the Object Base or Generic Shape Base.

Ratio features are usually measurements independent of an objects shape, orientation and size, for example, the compactness ratio (area/perimeter^2), thinness ratio (4π/compactness ratio), etc. In these cases the minimum absolute difference error rule is used for the comparison, which is based on the Max-Min square error principle. It can be described as follows:

Suppose

$\mathbf{x} = (x^1, x^2, ..., x^k)$ is the ratio feature vector of the object or blob to be verified,

$\mathbf{x}_b = (x_b^1, x_b^2, x_b^k)$ is the ratio feature vector of the model in the Generic Shape Base or Object Base,

if $| \mathbf{x} - \mathbf{x}_b | < \delta$

then $\mathbf{x}$ matches $\mathbf{x}_b$

(δ is a given error allowance which can be obtained experimentally.)

6.4.3 Reasoning Strategies

There are basically two reasoning strategies applied to most problem-solving tasks: **forward reasoning** and **backward reasoning.**

Forward reasoning is data driven and works from the known facts to a conclusion. In machine vision system, when the query is to ask what the given scene is, or what kind of object it contains", then forward reasoning is applied.

Backward reasoning is goal driven process. It breaks down the main problem, or goal, into a set of subproblems, or subgoals. The problem is solved by solving simpler subproblems. In the machine vision system, if the query is to check whether a given object name is in the scene, the system uses backward reasoning.

The Inference Engine used in this project is furnished with more reasoning power by incorporating a feedback function. That is, matching information is used to guide the image processing routine. This design consideration was motivated by the problem that exists in the merging arc and merging lines routines.

As was stated in Chapter 5, errors occur during the detection of dominant points. When the primitive is small, the type of primitive is easily mistaken. Conversely, a small line which is part of line might be regarded as a arc . A small arc which is part of arc might be thought of as a line, etc. This situation is illustrated in Figure 6-9.

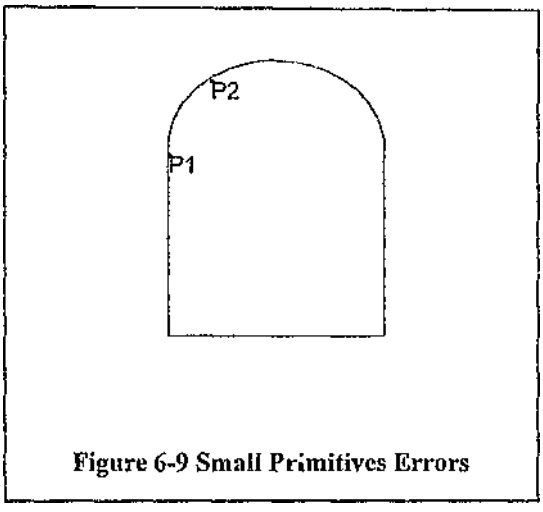


Figure 6-9 Small Primitives Errors

The problems are solved by setting up merging rules as follows:

(1) If the matching result from Blob-Verification is that:

the number of lines is matched, but

the number of arcs is bigger than that of the generic blob,

Then the smallest arc will be merged with the neighbouring lines or arc.

(2) If the matching result from Blob-Verification is that:

the number of arcs is matched or equal to zero, but

the number of lines is more than that of the generic blob,

Then the smallest lines will be merged with the neighbouring lines or arc.

The Inference Engine disposes of the unmatched information from the blob verification results in three steps:

(1) Fire the appropriate merging rule, and feedback the rule action to the Image Processing module. The image will be processed according to the new rule. Updated results will be presented.

(2) If the blob verification still fails to match a proper generic blob after the procedure detailed in step (1), the inference Engine will send a message (can be a trigger pulse) to change the camera direction. A renewed image scene will then be available for processing.

(3) If the blob verification routine still fails to find a good match, the Inference Engine will consult the user by asking whether this blob has to be added to the Generic Shape Base as a new generic blob.

The third procedure accomplishes the learning function for the machine vision system.

Because the feature errors result mainly from the low level image decomposition of the part, the feedback functions such as (1) and (2) in the above procedures, are not necessary for the presentation of information from the Object Verification routine.

If the Inference Engine fails to find a proper match as a result of a search through the Object Base, the user will be consulted to make a decision to classify it as a new object and add the definition to the Object Base.

6.4.4 Implementation

As can be appreciated from above discussion, the Inference Engine must have the ability to match facts and reasoning with the image scene. For the requirement of facts matching, two modules OBJ_VERIFY and BLOB_VERIFY were designed to implement the object verification and blob verification routines in this machine vision system

Another two modules were designed to process and feedback the information from the fact-matching functio. s.

OBJ_VERIFY searches through the Object Base to find a good match.

BLOB_VERIFY searches through the Generic Shape Base to match a generic blob.

B_RESULT presents the information from the blob verification routine.

O_RESULT processes the information from object verification.

6.5 Experimental Results and Discussions

1. Equilateral Triangle:

This first example shows the system identifying an equilateral triangle. The system is initiated with the high level image segmentation routine. The object is extracted as a single blob from the background with the edge, the centroid, and a bounding box as shown in Figure 6-10.

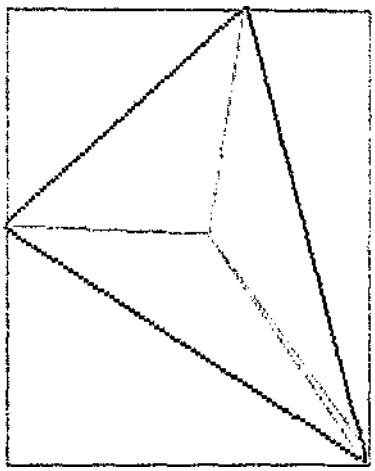


Figure 6-10 Eq-triangle

At the low level image processing phase, four initial arbitrary corner points are identified and the four primitives are extracted. Each primitive is displayed by its two endpoints P_1 and P_2 , the primitive type, the length of primitive and the angle of the primitive. in the data file. The type of the primitive is defined as:

1--A Point or small segments;

2--Lines;

3--or arcs;

4--others;

One primitive is identified as a small segment, and it is merged with the adjacent primitive (see the following data file).

Based on the primitives, the features of the blob are extracted. The inference engine uses them to search through the generic shape model base and the result is

displayed in the data file with both the information on the generic model and the results of the blob analysis.

```
File name :eq_tri.dat
Blob consist of ( 4 ) primitives.
Primitive[n]    P1              P2          Type    Length          Angle
primitive[1]    (364,134) (66,175)          2       300.807247      3.004867
primitive[2]    (66,175) (256,384)          2       282.455306      0.832981
primitive[3]    (256,384) (369,135)         2       273.441036      5.138411
primitive[4]    (369,135) (364,134)         1       5.099020        3.338988
After Small Segments merging:
Blob consist of ( 3 ) primitives.
Primitive[n]    P1              P2          Type    Length          Angle
primitive[1]    (369,135) (66,175)          2       305.628860      3.010338
primitive[2]    (66,175) (256,384)          2       282.455306      0.832981
primitive[3]    (256,384) (369,135)         2       273.441036      5.138411
Blob features are extracted as
Numbers of Lines: 3
Numbers of ARCs: 0
Numbers of Lines with equal length: 3
Numbers of equal angles: 2
Numbers of Angles =90 : 0
Numbers of Angles >90 : 0
Numbers of Angles <90 : 3
Area = 36864
Gravity Center ( 232 , 233 )
Perimeter =  947
Compactness Ratio = 24.378798
Thinness Ratio = 0.515463

Generic Shape Base:
Generic Shape No:8 EQ_TRIANGLE
Line Number: 3          Arc Number: 0
Equal line number: 3    Equal Angle Number: 3
Number of angle with 90: 0      Big than 90: 0    Less than 90: 3
Compact Ratio= 20.780001        Thinness Ratio= 0.605000
Area= -1 Perimeter= -1    Gravity Center:(-1, -1)

Blob Analysis Results:
Blob[2] Supposed name: EQ_TRIANGLE
Line Number: 3          Arc Number: 0
Equal line number: 3    Equal Angle Number: 2
Number of angle with 90: 0      Big than 90: 0    Less than 90: 3
Compact Ratio= 24.378798        Thinness Ratio= 0.515463
Area= 36864 Perimeter= 947      Gravity Center:(232, 233)

Object Inference result:
Number of Object:1
Object Analysis Results:
Supposed Object name:EQ_TRIANGLE
```

2. Isosceles_Triangle:

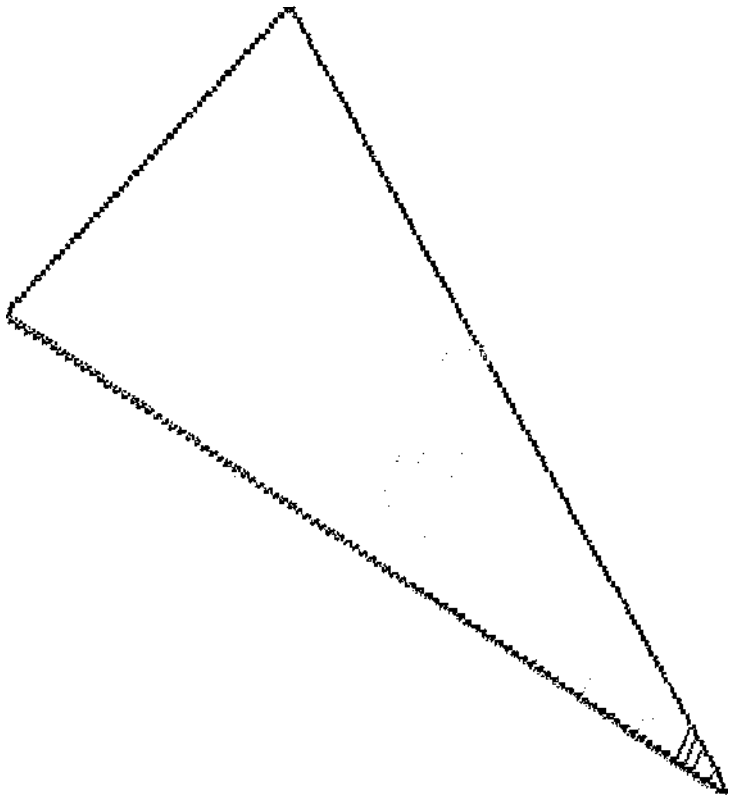


Figure 6-11 Iso_triangle

This example illustrates that blank points have occurred during the line verification process, the results of the analysis shows that the system can handle this situation successfully.

File name :iso_tri.dat
Blob consist of (4) primitives.

Primitive[n]	P1	P2	Type	Length	Angle
primitive[1]	(204,78)	(74,222)	2	194.000000	2.305144
primitive[2]	(74,222)	(340,296)	2	276.101431	0.271335
primitive[3]	(340,296)	(347,294)	1	7.280110	6.004886
primitive[4]	(347,294)	(204,78)	2	259.046328	4.127598

After Small Segments merging:
Blob consist of (3) primitives.

Primitive[n]	P1	P2	Type	Length	Angle
primitive[1]	(204,78)	(74,222)	2	194.000000	2.305144
primitive[2]	(74,222)	(340,296)	2	276.101431	0.271335
primitive[3]	(340,296)	(204,78)	2	256.943574	4.154615

Blob features are extracted as
Numbers of Lines: 3
Numbers of ARCs: 0
Numbers of Lines with equal length: 2
Numbers of equal angles: 0
Numbers of Angles =90 : 0
Numbers of Angles >90 : 0
Numbers of Angles <90 : 3
Area = 25724
Gravity Center (211 , 201)
Perimeter = 840
Compactness Ratio = 27.470922
Thinness Ratio = 0.457443

Generic Shape Base:
Generic Shape No:7 ISO_TRIANGLE
Line Number: 3 Arc Number: 0
Equal line number: 2 Equal Angle Number: 2
Number of angle with 90: 0 Big than 90: -1 Less than 90: -1
Compact Ratio= -1.000000 Thinness Ratio= -1.000000
Area= -1 Perimeter= -1 Gravity Center:(-1, -1)

Blob Analysis Results:
Supposed name: ISO_TRIANGLE
Line Number: 3 Arc Number: 0
Equal line number: 2 Equal Angle Number: 0
Number of angle with 90: 0 Big than 90: 0 Less than 90: 3
Compact Ratio= 27.470922 Thinness Ratio= 0.457443
Area= 25724 Perimeter= 840 Gravity Center:(211, 201)

Object Inference result:
Number of Object:1
Object Analysis Results:
Supposed Object name:ISO_TRIANGLE

3. Circle

This example illustrates the system identifying a circle. The results in the data file show that there are differences on compactness ratios between the practical computation and the ideal situation.

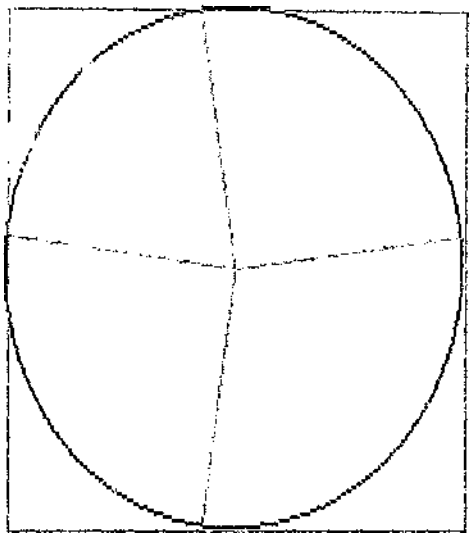


Figure 6-12 Circle

File name :circle.dat
Blob features are extracted as
Numbers of Lines: 0
Numbers of ARCs: 0
Numbers of Lines with equal length: 0
Numbers of equal angles: 0
Numbers of Angles =90 : 0
Numbers of Angles >90 : 0
Numbers of Angles <90 : 0
Area = 37910
Gravity Center (276 , 252)
Perimeter = 647
Compactness Ratio = 11.044684
Thinness Ratio = 1.137775

Generic Shape Base:
Generic Shape No:1 CIRCLE
Line Number: 0 Arc Number: 0
Equal line number: 0 Equal Angle Number: 0
Number of angle with 90: 0 Big than 90: 0 Less than 90: 0
Compact Ratio= 12.566371 Thinness Ratio= 1.000000
Area= 45238 Perimeter= 753 Gravity Center:(768, 768)

Blob Analysis Results:
Blob[2] Supposed name: CIRCLE
Line Number: 0 Arc Number: 0
Equal line number: 0 Equal Angle Number: 0
Number of angle with 90: 0 Big than 90: 0 Less than 90: 0
Compact Ratio= 11.044684 Thinness Ratio= 1.137775

Area= 37910 Perimeter= 647 Gravity Center:(276, 252)

Object Inference result:
Number of Object:1
Object Analysis Results:
Supposed Object name:CIRCLE

4. Square

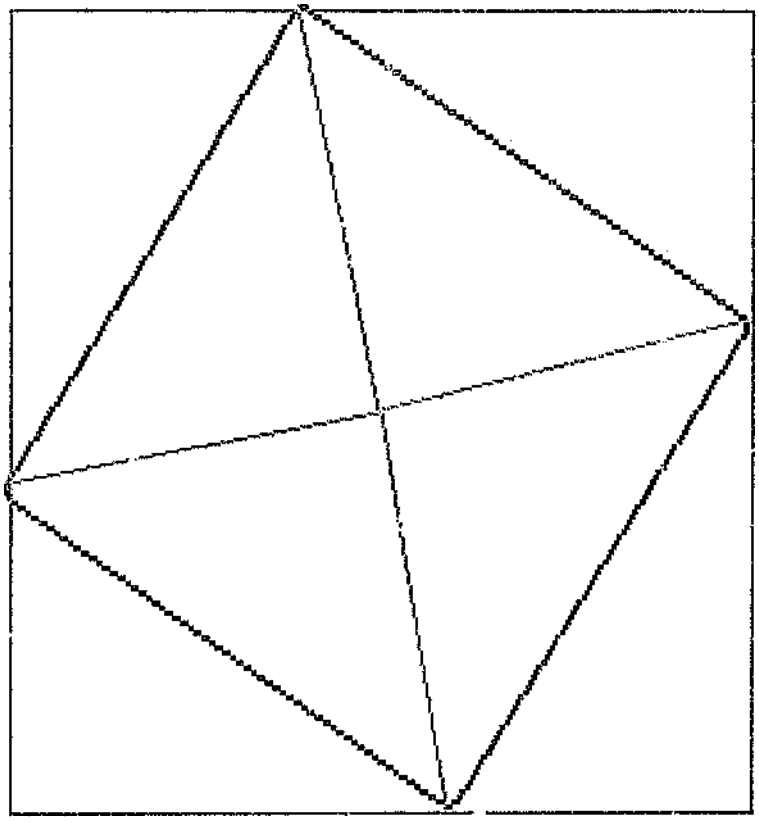


Figure 6-13 Square

This example illustrates that the system can handle the contradictory situation that has arisen from the image digitization error. The extracted features of the inspected object displayed in the following data file, show that both the number of

lines with same length and the number of angles of 90° are four, the number of equal angles are two. The inference uses rules which tolerate this error and correctly identify the shape.

File name :sqs.dat
Blob consist of (4) primitives.

Primitive[n]	P1	P2	Type	Length	Angle
primitive[1]	(276,101)	(86,211)	2	219.544984	2.616797
primitive[2]	(86,211)	(212,382)	2	212.407627	0.935770
primitive[3]	(212,382)	(407,268)	2	225.878286	5.754155
primitive[4]	(407,268)	(276,101)	2	212.249853	4.047214

Blob features are extracted as
Numbers of Lines: 4
Numbers of ARCs: 0
Numbers of Lines with equal length: 4
Numbers of equal angles: 2
Numbers of Angles =90 : 4
Numbers of Angles >90 : 0
Numbers of Angles <90 : 0
Area = 48606
Gravity Center (248 , 243)
Perimeter = 994
Compactness Ratio = 20.357157
Thinness Ratio = 0.617295

Generic Shape Base:
Generic Shape No:2 SQUARE
Line Number: 4 Arc Number: 0
Equal line number: 4 Equal Angle Number: 4
Number of angle with 90: 4 Big than 90: 0 Less than 90: 0
Compact Ratio= 16.000000 Thinness Ratio= 0.785400
Area= 65536 Perimeter= 1024 Gravity Center:(768, 768)

Blob Analysis Results:
Blob[2] Supposed name: SQUARE
Line Number 4 Arc Number: 0
Equal line ber: 4 Equal Angle Number: 2
Number gt with 90: 4 Big than 90: 0 Less than 90: 0
Compact Ratio= 20.357157 Thinness Ratio= 0.617295
Area= 48606 Perimeter= 994 Gravity Center:(248, 243)

Object Inference result:
Number of Object:1
Object Analysis Results:
Supposed Object name:SQUARE

5. Rectangle

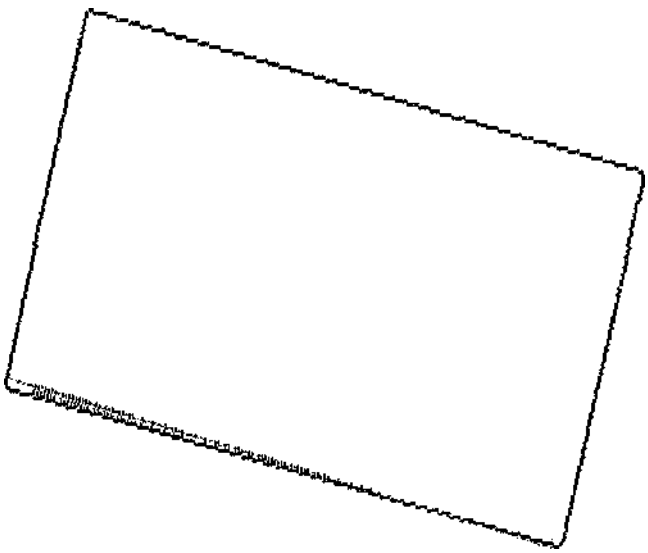


Figure 6-14 Rectangle

This example illustrates how the line verification searching scheme works for the situation that has arisen in Figure 6-14. The analysis also shows that the system can deal with the contradictory situation that has occurred during the blob inference phase.

File name :rectb.dat
Blob consist of (4) primitives.

Primitive[n]	P1	P2	Type	Length	Angle
primitive[1]	(268,85)	(48,135)	2	225.610283	2.918116
primitive[2]	(48,135)	(146,470)	2	349.040112	1.286200
primitive[3]	(146,470)	(371,415)	2	231.624696	6.043442
primitive[4]	(371,415)	(268,85)	2	345.700738	4.409849

Blob features are extracted as
Numbers of Lines: 4
Numbers of ARCs: 0
Numbers of Lines with equal length: 2
Numbers of equal angles: 4
Numbers of Angles =90 : 4
Numbers of Angles >90 : 0
Numbers of Angles <90 : 0
Area = 81832
Gravity Center (5 , 73)
Perimeter = 1255

Compactness Ratio = 19.273878
Thinness Ratio = 0.651990

Generic Shape Base:
Generic Shape No:3 RECTANGLE
Line Number: 4 Arc Number: 0
Equal line number: 2 Equal Angle Number: 4
Number of angle with 90: 4 Big than 90: 0 Less than 90: 0
Compact Ratio= -1.000000 Thinness Ratio= -1.000000
Area= 40960 Perimeter= 832 Gravity Center:(768, 768)

Blob Analysis Results:
Blob[2] Supposed name: RECTANGLE
Line Number: 4 Arc Number: 0
Equal line number: 2 Equal Angle Number: 4
Number of angle with 90: 4 Big than 90: 0 Less than 90: 0
Compact Ratio= 19.273878 Thinness Ratio= 0.651990
Area= 81832 Perimeter= 1255 Gravity Center:(5, 73)

Object Inference result:
Number of Object:1
Object Analysis Results:
Supposed Object name:RECTANGLE

6. Parallelogram

This example illustrates two concepts: firstly dealing with lines that have become disjointed during digitization, and secondly searching for new arbitrary corner points.

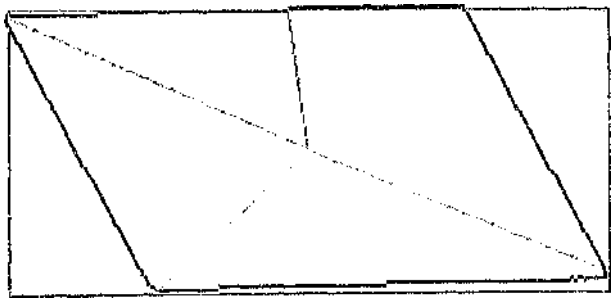


Figure 6-15 Parallelogram 1

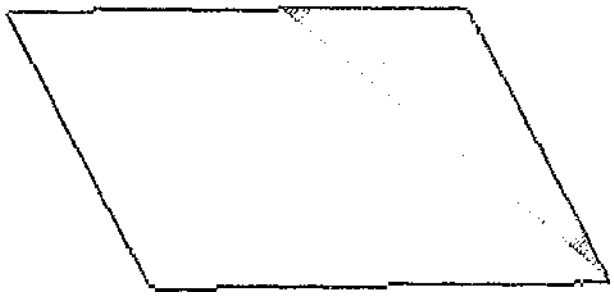


Figure 6-16 Parallelogram 2

In the Figure 6-16, the upper parallel line is digitized as three segments. The initial arbitrary point is located at the second one. The data file shows that five straight lines are identified (primitive type =2) at the first phase of processing. Figure 6-16 shows that a new corner point is identified during the line verification process.

After the line merging operation, four primitives are extracted .

File name :pl.dat

Blob consist of (5) primitives.

Primitive[n]	P1	P2	Type	Length	Angle
primitive[1]	(92,65)	(88,235)	2	170.047052	1.594321
primitive[2]	(88,235)	(89,340)	2	105.004762	1.570796
primitive[3]	(89,340)	(248,425)	2	180.294204	0.490936
primitive[4]	(248,425)	(259,155)	2	270.223981	4.753107
primitive[5]	(259,155)	(92,65)	2	189.707670	3.635891

After line merging:

Blob consist of (4) primitives.

Primitive[n]	P1	P2	Type	Length	Angle
primitive[1]	(92,65)	(89,340)	2	275.016363	1.570796
primitive[2]	(89,340)	(248,425)	2	180.294204	0.490936
primitive[3]	(248,425)	(259,155)	2	270.223981	4.753107
primitive[4]	(259,155)	(92,65)	2	189.707670	3.635891

Blob features are extracted as
Numbers of Lines: 4
Numbers of ARCs: 0
Numbers of Lines with equal length: 2
Numbers of equal angles: 2
Numbers of Angles =90 : 0
Numbers of Angles >90 : 2
Numbers of Angles <90 : 2
Area = 46922

Gravity Center (174 , 247)
Perimeter = 681
Compactness Ratio = 9.889498
Thinness Ratio = 1.270678

Generic Shape Base:
Generic Shape No:11 PARALLEL
Line Number: 4 Arc Number: 0
Equal line number: 2 Equal Angle Number: 2
Number of angle with 90: 0 Big than 90: 2 Less than 90: 2
Compact Ratio= -1.000000 Thinness Ratio= -1.000000
Area= -1 Perimeter= -1 Gravity Center:(-1, -1)

Blob Analysis Results:
Blob[2] Supposed name: PARALLEL
Line Number: 4 Arc Number: 0
Equal line number: 2 Equal Angle Number: 2
Number of angle with 90: 0 Big than 90: 2 Less than 90: 2
Compact Ratio= 9.889498 Thinness Ratio= 1.270678
Area= 46922 Perimeter= 681 Gravity Center:(174, 247)

Object Inference result:
Number of Object:1
Object Analysis Results:
Supposed Object name:PARALLEL

7. Six_Gon

This example also illustrates how the system finds the new corner points with the identified initial arbitrary corner points. The four initial corner points are located at the bounding box and the other two corner points are identified by the line verification searching scheme.

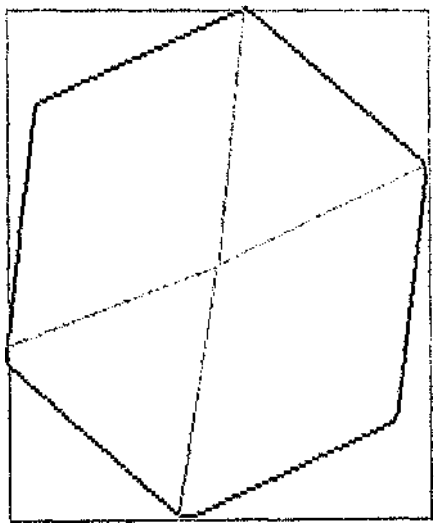


Figure 6-17 Six-gon 1

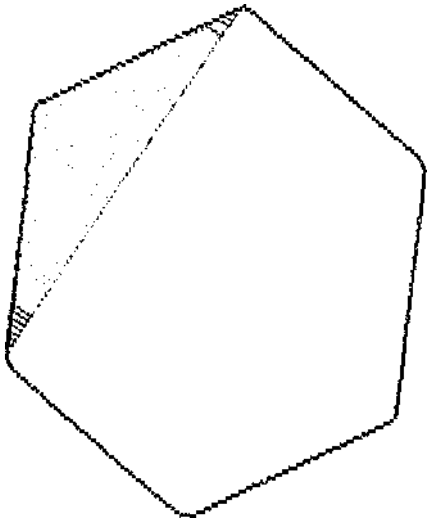


Figure 6-18 Six-gon 2

File name :sixg.dat

Blob consist of (6) primitives.

Primitive[n]	P1	P2	Type	Length	Angle

primitive[1]	(278,124) (134,153)	2	146.891116	2.942862
primitive[2]	(134,153) (86,276)	2	132.034037	1.942864
primitive[3]	(86,276) (188,383)	2	147.827602	0.809317
primitive[4]	(188,383) (339,355)	2	153.574086	6.099837
primitive[5]	(339,355) (389,225)	2	139.283883	5.079563
primitive[6]	(389,225) (278,124)	2	150.073315	3.879856

Blob features are extracted as
Numbers of Lines: 6
Numbers of ARCs: 0
Numbers of Lines with equal length: 6
Numbers of equal angles: 6
Numbers of Angles =90 : 0
Numbers of Angles >90 : 6
Numbers of Angles <90 : 0
Area = 56028
Gravity Center (239 , 255)
Perimeter = 950
Compactness Ratio = 16.140715
Thinness Ratio = 0.778551

Generic Shape Base:
Generic Shape No:4 SIX_GON
Line Number: 6 Arc Number: 0
Equal line number: 6 Equal Angle Number: 6
Number of angle with 90: 0 Big than 90: 6 Less than 90: 0
Compact Ratio= 13.860000 Thinness Ratio= 0.906600
Area= 25980 Perimeter= 600 Gravity Center:(768, 768)

Blob Analysis Results:
Blob[2] Supposed name: SIX_GON
Line Number: 6 Arc Number: 0
Equal line number: 6 Equal Angle Number: 6
Number of angle with 90: 0 Big than 90: 6 Less than 90: 0
Compact Ratio= 16.140715 Thinness Ratio= 0.778551
Area= 56028 Perimeter= 950 Gravity Center:(239, 255)

Object Inference result:
Number of Object:1
Object Analysis Results:
Supposed Object name:SIX_GON

8. Circle_Circle

This example shows the system identifying an object with a hole. The object is extracted as two labeled blobs with a bounding box imposed on each of them, displayed in Figure 6-19.

Each blob is processed separately and identified through the blob verification scheme. The results are displayed in the data file. After all the blobs are identified,

the object verification function give an inference which is displayed at the end of the data file.

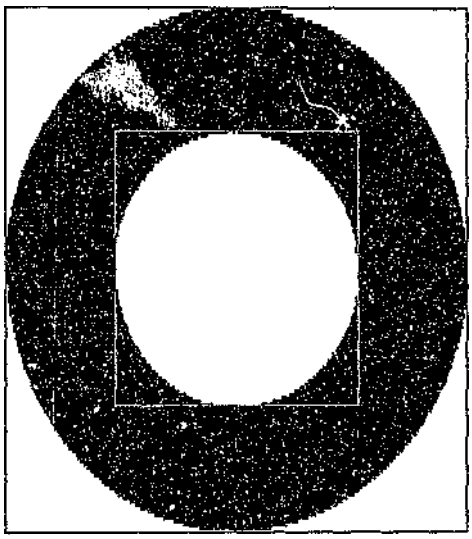


Figure 6-19 Circle_Circle

File name :cl_cl.dat
Blob features are extracted as
Numbers of Lines: 0
Numbers of ARCs: 0
Numbers of Lines with equal length: 0
Numbers of equal angles: 0
Numbers of Angles =90 : 0
Numbers of Angles >90 : 0
Numbers of Angles <90 : 0
Area = 68220
Gravity Center (244 , 219)
Perimeter = 896
Compactness Ratio = 11.773146
Thinness Ratio = 1.067376

Generic Shape Base:
Generic Shap No:1 CIRCLE
Line Number: 0 Arc Number: 0
Equal line number: 0 Equal Angle Number: 0
Number of angle with 90: 0 Big than 90: 0 Less than 90: 0
Compact Ratio= 12.566371 Thinness Ratio= 1.000000
Area= 45238 Perimeter= 753 Gravity Center:(768, 768)

Blob Analysis Results:
Blob[2] Supposed name: CIRCLE
Line Number: 0 Arc Number: 0
Equal line number: 0 Equal Angle Number: 0
Number of angle with 90: 0 Big than 90: 0 Less than 90: 0
Compact Ratio= 11.773146 Thinness Ratio= 1.067376
Area= 68220 Perimeter= 896 Gravity Center:(244, 219)

Blob features are extracted as
Numbers of Lines: 0
Numbers of ARCs: 0
Numbers of Lines with equal length: 0
Numbers of equal angles: 0
Numbers of Angles =90 : 0
Numbers of Angles >90 : 0
Numbers of Angles <90 : 0
Area = 19194
Gravity Center (245 , 219)
Perimeter = 446
Compactness Ratio = 10.377618
Thinness Ratio = 1.210911

Generic Shape Base:
Generic Shape No:1 CIRCLE
Line Number: 0 Arc Number: 0
Equal line number: 0 Equal Angle Number: 0
Number of angle with 90: 0 Big than 90: 0 Less than 90: 0
Compact Ratio= 12.566371 Thinness Ratio= 1.000000
Area= 45238 Perimeter= 753 Gravity Center:(768, 768)

Blob Analysis Results:
Blob[3] Supposed name: CIRCLE
Line Number: 0 Arc Number: 0
Equal line number: 0 Equal Angle Number: 0
Number of angle with 90: 0 Big than 90: 0 Less than 90: 0
Compact Ratio= 10.377618 Thinness Ratio= 1.210911
Area= 19194 Perimeter= 446 Gravity Center:(245, 219)

Object Inference result:
Number of Object:1

Object Base:
Object Shape No:4 CIRCLE_CIRCLE
Object outline shape is CIRCLE
It consists of 1 Hole( ):
Area Ratio of Hole 1 = 0.250000
The hole's generic no = 1 is CIRCLE

Object Analysis Results:
Object[1] Supposed name: CIRCLE_CIRCLE
Object outline shape is CIRCLE
It consists of 1 Hole(s):
Area Ratio of Hole 1 = 0.281354
The hole's generic no = 1 is CIRCLE

9. Square_Circle

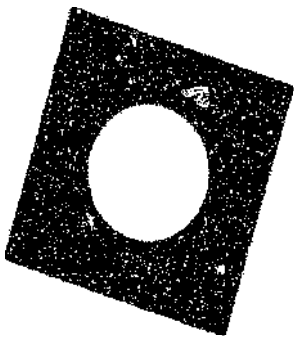


Figure 6-20 Square-Circle

This example illustrates the system identifying an object with a hole.

File name :sq_cl.dat
Blob consist of (4) primitives.

Primitive[n]	P1	P2	Type	Length	Angle
primitive[1]	(338,69)	(46,155)	2	304.401051	2.855170
primitive[2]	(46,155)	(144,416)	2	278.792037	1.211606
primitive[3]	(144,416)	(441,325)	2	310.628395	5.985870
primitive[4]	(441,325)	(338,69)	2	279.943835	4.329864

Blob features are extracted as
Numbers of Lines: 4
Numbers of ARCs: 0
Numbers of Lines with equal length: 4
Numbers of equal angles: 3
Numbers of Angles =90 : 4
Numbers of Angles >90 : 0
Numbers of Angles <90 : 0
Area = 87350
Gravity Center (52 , 51)
Perimeter = 1290
Compactness Ratio = 19.078146
Thinness Ratio = 0.658679

Generic Shape Base:
Generic Shap No:2 SQUARE
Line Number: 4 Arc Number: 0
Equal line number: 4 Equal Angle Number: 4
Number of angle with 90: 4 Big than 90: 0 Less than 90: 0
Compact Ratio= 16.000000 Thinness Ratio= 0.785400
Area= 65536 Perimeter= 1024 Gravity Center:(768, 768)

Blob Analysis Results:
Blob[2] Supposed name: SQUARE

Line Number: 4 Arc Number: 0
Equal line number: 4 Equal Angle Number: 3
Number of angle with 90: 4 Big than 90: 0 Less than 90: 0
Compact Ratio= 19.078146 Thinness Ratio= 0.658679
Area= 87350 Perimeter= 1290 Gravity Center:(52, 51)

Blob features are extracted as
Numbers of Lines: 0
Numbers of ARCs: 0
Numbers of Lines with equal length: 0
Numbers of equal angles: 0
Numbers of Angles =90 : 0
Numbers of Angles >90 : 0
Numbers of Angles <90 : 0
Area = 18974
Gravity Center (245 , 244)
Perimeter = 444
Compactness Ra = 10.412564
Thinness Rat 6847

Generic Shape ?:
Generic Shap No:1 CIRCLE
Line Number: 0 Arc Number: 0
Equal line number: 0 Equal Angle Number: 0
Number of angle with 90: 0 Big than 90: 0 Less than 90: 0
Compact Ratio= 12.566371 Thinness Ratio= 1.000000
Area= 45238 Perimeter= 753 Gravity Center:(768, 768)

Blob Analysis Results:
Blob[3] Supposed name: CIRCLE
Line Number: 0 Arc Number: 0
Equal line number: 0 Equal Angle Number: 0
Number of angle with 90: 0 Big than 90: 0 Less than 90: 0
Compact Ratio= 10.412564 Thinness Ratio= 1.206847
Area= 18974 Perimeter= 444 Gravity Center:(245, 244)

Object Inference result:
Number of Object:1

Object Base:
Object Shap No:1 SQUARE_CIRCLE
Object outline shape is SQUARE
It consists of 1 Hole(s):
Area Ratio of Hole 1 = 0.196000
The hole's generic no = 1 is CIRCLE
Object Analysis Results:
Object[1] Supposed name: SQUARE_CIRCLE
Object outline shape is SQUARE
It consists of 1 Hole(s):
Area Ratio of Hole 1 = 0.217218
The hole's generic no = 1 is CIRCLE

10. Six_Circle

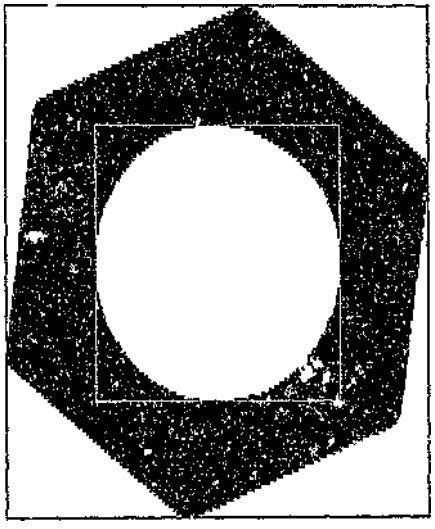


Figure 6-21 Six_gon-Circle

This example also illustrates how the system successfully identifies an object with a hole.

File name :six_cl.dat
Blob consists of (6) primitives.

Primitive[n]	P1	P2	Type	Length	Angle
primitive[1]	(300,96)	(155,114)	2	146.112970	3.018087
primitive[2]	(155,114)	(94,238)	2	138.191896	2.027972
primitive[3]	(94,238)	(190,349)	2	146.754898	0.857735
primitive[4]	(190,349)	(345,327)	2	156.553505	6.142192
primitive[5]	(345,327)	(401,201)	2	137.884009	5.130613
primitive[6]	(401,201)	(300,96)	2	145.691455	3.946406

Blob features are extracted as
Numbers of Lines: 6
Numbers of ARCs: 0
Numbers of Lines with equal length: 6
Numbers of equal angles: 6
Numbers of Angles =90 : 0
Numbers of Angles >90 : 6
Numbers of Angles <90 : 0
Area = 56190
Gravity Center (250 , 224)
Perimeter = 947

Compactness Ratio = 15.966578
Thinness Ratio = 0.787042

Generic Shape Base:
Generic Shap No:4 SIX_GON
Line Number: 6 Arc Number: 0
Equal line number: 6 Equal Angle Number: 6
Number of angle with 90: 0 Big than 90: 6 Less than 90: 0
Compact Ratio= 13.860000 Thinness Ratio= 0.906600
Area= 25980 Perimeter= 600 Gravity Center:(768, 768)

Blob Analysis Results:
Blob[2] Supposed name: SIX_GON
Line Number: 6 Arc Number: 0
Equal line number: 6 Equal Angle Number: 6
Number of angle with 90: 0 Big than 90: 6 Less than 90: 0
Compact Ratio= 15.966578 Thinness Ratio= 0.787042
Area= 56190 Perimeter= 947 Gravity Center:(250, 224)

Blob features ... extracted as
Numbers of Lines: 0
Numbers of ARCs: 0
Numbers of Lines with equal length: 0
Numbers of equal angles: 0
Numbers of Angles =90 : 0
Numbers of Angles >90 : 0
Numbers of Angles <90 : 0
Area = 19138
Gravity Center (250 , 224)
Perimeter = 441
Compactness Ratio = 10.194848
Thinness Ratio = 1.232620

Generic Shape Base:
Generic Shap No:1 CIRCLE
Line Number: 0 Arc Number: 0
Equal line number: 0 Equal Angle Number: 0
Number of angle with 90: 0 Big than 90: 0 Less than 90: 0
Compact Ratio= 12.566371 Thinness Ratio= 1.000000
Area= 45738 Perimeter= 753 Gravity Center:(768, 768)

Blob Analysis Results:
Blob[3] Supposed name: CIRCLE
Line Number: 0 Arc Number: 0
Equal line number: 0 Equal Angle Number: 0
Number of angle with 90: 0 Big than 90: 0 Less than 90: 0
Compact Ratio= 10.194848 Thinness Ratio= 1.232620
Area= 19138 Perimeter= 441 Gravity Center:(250, 224)

Object Inference result:
Number of Object:1

Object Base:
Object Shap No:5 SIX_CIRCLE
Object outline shape is SIX_GON

It consists of 1 Hole(s):
Area Ratio of Hole 1 = 0.302000
The hole's generic no = 1 is CIRCLE
Object Analysis Results:
Object[1] Supposed name: SIX_CIRCLE
Object outline shape is SIX_GON
It consists of 1 Hole(s):
Area Ratio of Hole 1 = 0.340594
The hole's generic no = 1 is CIRCLE

10. Window Shape

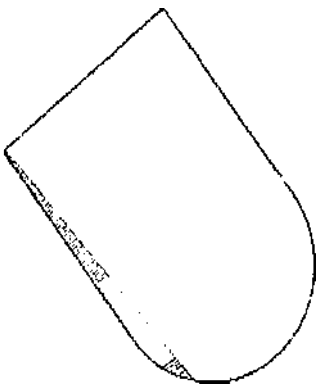


Figure 6-22 Window 1

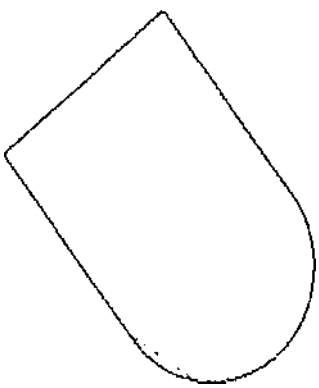


Figure 6-23 Window 2

This example shows how the system identifies an object consisting of arcs and how it deals with the problem of merging the primitives correctly.

Figure 6-22 shows the leftmost primitive connected by two initial arbitrary points which can be either a line or an arc. The line verification scheme breaks the primitive into two primitives. Figure 6-23 shows the smaller primitive is identified as either an arc or a line. The merging routines merge it with the adjacent arc. The results are recorded in the data file.

File name :wind.dat
Blob consists of (5) primitives.

Primitive[n]	P1	P2	Type	Length	Angle
primitive[1]	(180,61)	(38,219)	2	212.433519	2.302912
primitive[2]	(38,219)	(282,372)	2	288.001736	0.560072
primitive[3]	(282,372)	(411,255)	3	200.800896	5.171582
primitive[4]	(411,255)	(365,188)	4	227.437904	3.739351
primitive[5]	(365,188)	(180,61)	2	227.437904	3.739351

After merging Arcs:
Blob consists of (4) primitives.

Primitive[n]	P1	P2	Type	Length	Angle
primitive[1]	(180,61)	(38,219)	2	212.433519	2.302912
primitive[2]	(38,219)	(282,372)	2	288.001736	0.560072
primitive[3]	(282,372)	(365,188)	3	201.853908	5.136146
primitive[4]	(365,188)	(180,61)	2	227.437904	3.739351

Blob features are extracted as
Numbers of Lines: 3
Numbers of ARCs: 1
Numbers of Lines with equal length: 2
Numbers of equal angles: 0
Numbers of Angles =90 : 1
Numbers of Angles >90 : 0
Numbers of Angles <90 : 2
Area = 67908
Gravity Center (239 , 232)
Perimeter = 1147
Compactness Ratio = 19.394623
Thinness Ratio = 0.647931

Generic Shape Base:
Generic Shap No:10 WINDOW
Line Number: 3 Arc Number: 1
Equal line number: -1 Equal Angle Number: 2
Number of angle with 90: 2 Big than 90: 0 Less than 90: 0
Compact Ratio= -1.000000 Thinness Ratio= -1.000000
Area= -1 Perimeter= -1 Gravity Center:(-1, -1)

Blob Analysis Results:
Blob[2] Supposed name: WINDOW
Line Number: 3 Arc Number: 1
Equal line number: 2 Equal Angle Number: 0
Number of angle with 90: 1 Big than 90: 0 Less than 90: 2
Compact Ratio= 19.394623 Thinness Ratio= 0.647931
Area= 67908 Perimeter= 1147 Gravity Center:(239, 232)

7. COMPARISON AND DISCUSSION

This chapter commences with analyses and comparisons of existing machine vision systems in industrial environments. This is followed by a summary of the results of the work carried out on this project including a description of the achievements and failures. Finally, suggestions are made for future work that should be carried out on the system.

7.1 Comparison with Other Systems

For the last two decades, many machine vision systems have been designed and implemented either partially or completely. Among the industrial applications, only a few are explicitly concerned with the generality of industrial parts. Most of them were designed for a specific task.

Holland, Rossol and Ward (1979) of the General Motors Research Laboratories designed a vision-based robot system that picks up parts randomly placed on a moving conveyor belt. In their system, only simple global geometric features such as orientation are computed for the position determination. If a new part was introduced, the vision system needed to be reprogrammed. Their work though, can be regarded as the classical model for most machine vision system in industrial environments.

Vamos (1977) dealt with industrial objects and machine parts in a more extensive manner. He claimed that a grammatical method is the only reasonable way for the problem to be defined for the recognition of industrial parts, especially machined

parts. Syntactic methods have been heavily developed but mostly constrained within the theoretical fields. He was probably the first to apply these methods to build a working system for an industrial application.

In his system, the industrial objects were described by the edge primitives which were classified in four types: the quoted straight lines, arcs, nodes, further undefined types being uncertain entities. A context-free grammar was used to build up the machined parts from the picture primitives. All the results were made up of a very detailed structural description of the whole scenery concerned.

His methodology of object description is similar to that in the system proposed here: that of using the boundary primitives to describe the machine parts. However, the low level image processing method is completely different. Vamos used a programmable laser-deflector to implement the contour-line detection. The system equipment consisted of a laser, an acousto-optical deflector with a resolution of 1000 points in each dimension and a number of photosensors mounted in different directions. If an edge is detected by the beam, at least one of the deflectors indicates a sharp change in the output signal.

It is felt that this feature extraction method which has a high requirement on system hardware, plus the side effects of an uneconomic, difficult implementation and the severe limitations on the environment in which the system can operate, pose severe restrictions for the use of this system.

Wu and Rodd (1992) explored the primitives extraction problems in an industrial environment and proposed a method for parts recognition which is a

combination of Gaussian-filter approach and an angle-detection scheme. The angle detection scheme is based on a series of defined criteria. However they did not proceed further to design an object recognition function

The similarity between the system developed here and that of Wu and Rodd, is that both have same application domain: i.e., flat machine-made industrial parts. The feature extraction method of our system is also similar to the method used by Wu and Rodd: that is using algorithms, instead of a hardware implementation. The difference is that ratios of segment length over the chord length between the two nodes are not used as the criteria for linearity or circularity. From the experience gained on this project it was concluded that image digitization using a computer can produce large errors in the computation of these ratios.

However, the most important difference between the two methods is that in the system developed for this project, feedback information from the recognition stage is used to adapt the image processing operations to the objects being analyzed.

The following statements can be made regarding the majority of existing machine vision systems in industrial applications:

(1) The image processing operation and the recognition operation are treated as separate tasks. It has been assumed that the analysis process of a machine vision system is either data-driven or model-driven. The data-driven procedures extract information related to light intensity variations in the image. and then interpret the information as a description of the objects. The model-driven procedure is a process

which searches for candidate models through the data base. We feel that this assumption needs to be justified. Building links between the two basic tasks in machine vision systems will accomplish the function of knowledge sharing within the machine vision system.

(2) The machine made parts have geometric regularity. This may explain why decomposition of objects into basic primitives is a reasonable solution to generate object descriptions in the machine vision system.

(3) Design of low level image processing algorithms should consider the factors for a future hardware implementation. Computational complexity is often the limiting factor for the data imaging process. Dedicated image processors are getting popular not only because their costs are decreasing, but also because they are more flexible for the generality of application tasks. Image processors are good at searching for points or for carrying out a transformation within a specific image domain. It is felt that a good algorithm for low level image data processing should take more appropriate point searching operations on the image domain instead of carrying out complex computations. Most image processing methods lack such a consideration.

7.2 Achievements of the new system

This programme of work has achieved the following goals:

(1) A working machine vision system which is capable of locating and identifying regular, flat, machine-made parts.

(2) Improving low level image processing by employing an interaction between the recognition stage and the image processing stage.

(3) Using knowledge-based techniques to organize the data and implement the recognition function.

(4) A shape description method for industrial parts.

As shown in Chapter 6, the system developed was successful in identifying and locating a range of industrial parts. Nineteen different machine-made parts, differing either in shape or size, were tested using the system. The recognition success rate was 100% in all cases. The results also showed that the system is independent of object orientation.

The use of information from the inference engine was a key point in accomplishing good results on the boundary primitive extraction. The lack of a good boundary primitive extraction process is a common weakness of existing machine vision systems and most image processing techniques.

Benefits are also reaped as a result of incorporating knowledge based techniques into the system; the most important of these advantages is the ease with which the system can be expanded. By placing the description of objects and generic blobs in separate data bases, simple updating can be implemented.

The shape description methods used in the system are based on a two level image decomposition scheme. At the low level, the generic shape is described by basic primitives. This description scheme was found to readily link with CAD/CAM systems, in which only limited, well-defined primitives elements are used.

7.3 Weaknesses of the system

Although the system that has been developed has been proved to be highly successful when regular geometric shapes are considered, there are limitations on the application domain.

(1) One limitation is when complex boundaries with frequent slope changes, such as occur in the 'saw-tooth' profile, are presented. The reason for the system failing to satisfactorily process this example of a complex edge may be as a result of the lack of definition of the description features for this type of feature. Special provision might well have to be introduced so that this kind of edge can be uniquely identified instead of using basic primitive segments.

(2) Objects overlapping: As the system uses a tree like structure to represent the blob nesting relationship, it fails to identify the scene where objects overlap.

(3) Objects touching each other: when objects are touching, the tendency is that they will be recognized as one object by this system. This problem can be resolved by defining the combination of generic shapes. Further study on the shape combinations is still needed.

(4) 3-D objects. To date, 3-D objects are considered to be outside of the scope of the application domain. At this stage, only a study of one view from the image scene, in which shadows are not present, has been considered. Although this can be perceived as a weakness, the majority of industrial machine vision applications are satisfied by a two-dimensional analysis.

7.4 Future Work

There are three areas where future work could improve the performance of the system.

The first area to be considered should be the exploitation of CAD/CAM techniques. The reason for this consideration is based on the following facts:

(1) Our application aims to deal with identifying components produced in the manufacturing environment, the characterization of manufactured parts is the main area of application for our system. Most industrial parts are designed using CAD/CAM systems, and it is a trend that industrial design is getting more dependent on CAD/CAM techniques.

(2) Although research on CAD/CAM techniques has been concentrated over the last two or three decades, it has however, been one of the fastest developing areas of computer-aided processes and has been promoted by particularly rapid developments in the associated hardware. Large amounts of resources, both in terms of funds and manpower have been invested in designing and building libraries of mechanical parts. The knowledge bases developed as a result of the application of CAD/CAM systems could prove to be a rich source of reference for use by machine vision systems. The integration of CAD/CAM libraries with the databases of a machine vision system offers the prospect of an immensely powerful machine vision system which will be invested with considerable intelligence.

(3) In CAD systems, very few primitives are used to build the mechanical parts, of these lines and circular arcs are the most popular choices. Our generic shape description scheme is compatible with this requirement. Adaptation between CAD systems and our machine vision system can be easily accomplished once the links between the two systems are established.

(4) Employment of CAD/CAM techniques on object formation techniques will shed some light on the problems highlighted in the previous section, namely those associated with objects touching and overlapping, and the visualization of 3-Dimensional scenes.

The second area in which system development is required is that of the recognition of moving objects. The system set-up has been designed in a manner that will easily allow this change to be made. However, the use of sequential scenes in the image processing and recognition stages still has to be studied.

Finally, the last area that needs to be considered is the introduction of more dominant features in the shape description scheme. The inclusion of these additional dominant features would be of considerable benefit when analysis of components is carried out that are typified by the 'saw tooth' like shapes, such as screw threads.

8. MACHINE VISION AND FUZZY SETS IN FRUIT CLASSIFICATION

8.1 INTRODUCTION

8.1.1 The Need For Automation of Fruit Inspection

South Africa is one of the largest fruit production countries in the world. Although some post-harvest handling and packaging of fresh market fruit has been extensively automated, the sorting and grading operation continues to be performed manually due to the need for intelligent interpretation in the grading process. Fruit packing houses are staffed with large numbers of workers, sorting out fruit which does not meet grade standards. It is an accepted fact that manual sorting is tedious, inconsistent and labour intensive. Owing to the sheer volume of product and the fact that subjective decisions have to be made rapidly, there is a large potential for human error. Automation of sorting and grading of fresh fruit has the potential to give an improvement in product quality, and in addition reduce labour costs.

8.1.2 Machine Vision in the Food Industry

Machine vision applied to non-contact inspection has found many applications in the industrial environment. Stimulated by the promise of maintaining accuracy and consistency, while eliminating the subjectivity of manual inspection, machine vision is the subject of increasing interest in food processing and agricultural

production industries. From the middle of the 1980s onwards a number of papers have been published on the use of machine vision in this sector, among these are (Bulley, MacDonald and Sziklai,1984), (Davison and Meyer,1984), (Sutton and Mclendon,1985), (Sarkar and Wolfe,1985), (Rehkugler and Throop,1987), (Westerlind,1988), and (Misra, Koermer and Shyy,1989).

Rapid developments in computer hardware technology and imaging devices has further promoted research into machine vision applications in the food and agricultural sectors during the present decade.

The following are examples of recent research: Miller and Delwiche (1991) developed a laboratory machine vision system to detect defects on fresh peaches. Ling and Searcy (1991) used machine vision to determine the optimum location for the removal of shrimp heads. Churchill, Bilsland and Cooper(1993) conducted experiments on comparison between machine vision and human measurements of seed dimensions. The results showed that there was a greater consistency for machine vision measurements over those obtained by human inspectors, plus there was a saving of over 30% in processing time.

Das and Evans (1992a,1992b) proposed the use of machine vision to automatically detect fertile eggs in the early stages after incubation. Their algorithms were based on analysis of gray level histogram of the eggs' image.

Goodrum and Elster (1992) built a machine vision system to commercial specifications using a PC-based computer. Through a surface analysis algorithm it was able to detect cracks in rotating eggs. The results showed the high success

rate, but these were dependent upon the size of the eggs and the system configuration.

Rigney, Brusewitz and Kranzler (1992) designed a statistical classifier for grading fresh market asparagus. The error rate in classification was between 8 and 42%.

Okamura, Delwiche and Thompson (1993) designed a machine vision system for grading raisins, in which a Bayers classifier was used. Six features were taken on the raisins. The grading results proved to be good when high quality raisins were used.

Simonton and Pease (1993) used machine vision to classify the major plant parts for the purpose of ornamental cutting. Their algorithm is orientation independent but had only a limited degree of effectiveness for the geranium cutting image.

Churchill, Bilsland and Cooper (1993) used machine vision to separate mixed lots of tall fescue and ryegrass seed. They took 10 physical properties of the seeds as dominant features and used a regression model for the classification algorithm. The average success rate for the process was between 72.64 and 83.8%.

The conclusion that can be drawn from the above examples is that these systems are all object dependent due to the complexity and differences in each items' appearance. All the processes are time consuming and most are still at laboratory research stage.

It is our opinion that the precision required for the expression of details in the boundary recognition stage sacrifices data processing time and limits the application of machine vision to practical industrial tasks. In fact, human beings

often perform their reasoning processes by not taking into account strict numerical quantities but instead focus their attention on categories.

The introduction of the concept of fuzzy sets into fruit inspection is a novel idea which has the effect of incorporating human intelligence into a machine vision system.

8.1.3 Aims of the Study

The aim of the study was to design and implement a machine vision system which would be capable of classifying a given fruit into different categories. The objectives of this project are:

- Develop an image processing algorithm capable of extracting and analyzing the descriptive features of a given fruit sample.
- Construct a fuzzy classifier for sorting fruit and identifying defects.

8.1.4 System Overview

To accomplish these goals, a machine vision system was designed that worked in three operational modes: namely the configuration mode, the learning mode and the classification mode.

In the learning mode, an image analysis was carried out and feature measurements taken on given samples of fruit to establish object templates. These templates were derived by performing statistical calculations on the descriptive features. The function of the learning mode is illustrated in Figure 8-1.

In the classification mode, the classification task is performed by matching the features extracted from the image with the object template. The operation of this mode is illustrated in the inner frame in Figure 8-1.

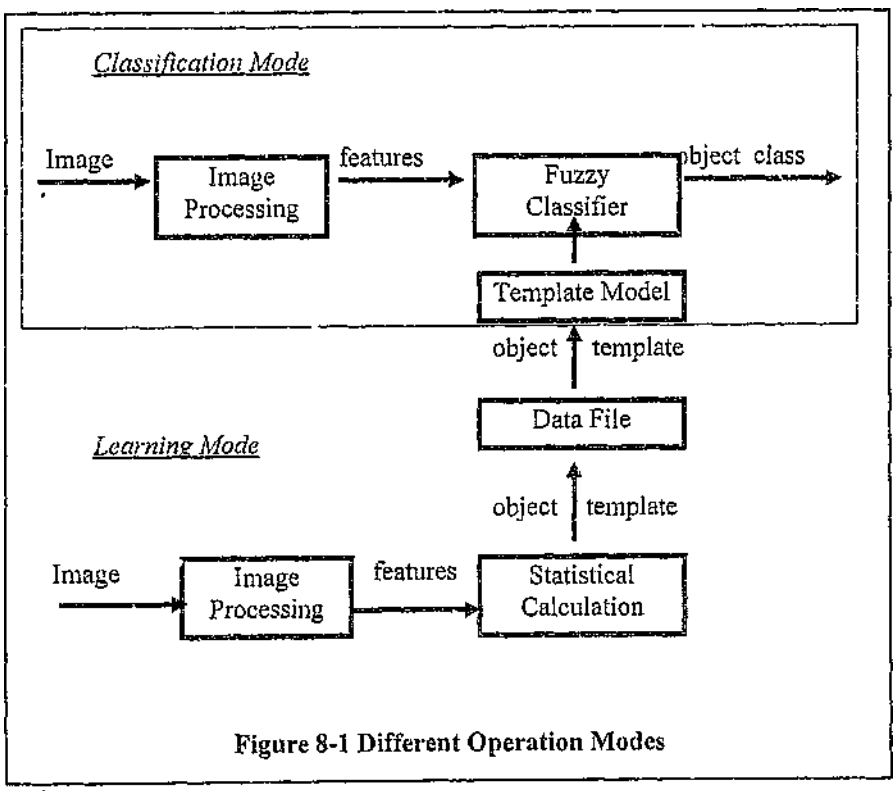


Figure 8-1 Different Operation Modes

In the configuration mode, the lighting condition is adjusted to avoid the shading problem.

The three modes involve four basic tasks: the image acquisition, the image segmentation, the feature extraction and template construction.

Following this section a full description of the image acquisition process is given in section 8.2. The system setup, memory management, coding method and shading correction are also discussed in this section.

Section 8.3 explains the method used to segment the object of interest from the background. The problem of noise removal is also discussed.

In Section 8.4, the issue of how to extract and compute the dominant features for the fruit classification is explored.

Section 8.5 deals with the construction of the fuzzy classifier. After an introduction to the concept of fuzzy sets and their application, the problem of constructing the membership function is discussed.

Section 8.6 explains how to classify the objects that result from the learning algorithms. Section 8.7 gives details on how to implement the program, and finally Section 8.8 presents the experimental results and draws appropriate conclusions.

8.2 IMAGE ACQUISITION

The first step in a machine vision system is to acquire a perfect digitized image. It is an important step since the image that is captured provides all the information

on which further image processing operations are based, the saying "rubbish in, rubbish out" is very pertinent to this process..

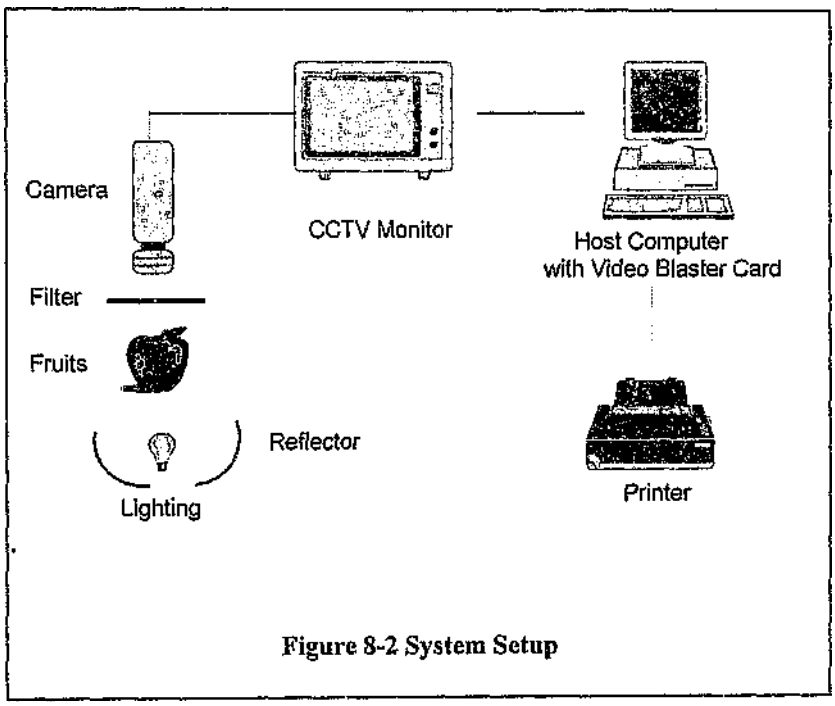


Figure 8-2 System Setup

This section conveys the basic information about the hardware of the machine vision system in the project, especially the Video Blaster frame grabber card. Emphasis is given to data management issues and coding techniques. The shading correction and lighting problems are also discussed.

8.2.1 Machine Vision System Setup

The machine system setup is illustrated in Figure 8-2. The system basically consists of a camera, a display monitor, an 80486 microcomputer and a Video Blaster Card.

The camera used to capture the image was chosen as a Panasonic WV-CL110e CCD camera due to its reasonable price. It was linked to a National WV-CM110a colour video monitor which was used to display the image. The output from the camera was connected to the video monitor and the monitor output was directed to the Video Blaster Card which was plugged into the mainframe slot of the microcomputer.

The Video Blaster Card provided from Creative Labs is a video overlay card that mixes video from a live video source with VGA graphics to be displayed on a VGA monitor. The computer monitor was configured to VGA graphics mode which supports screen resolutions of 640 by 480 pixels in 16 colours.

The algorithm was developed using functions provided by the Video Blaster Developer Kit. These functions were presented in the form of a resident driver called VBLSTDRV. Each function can be activated by calling its interrupt vector number at which it resides.

Microsoft C/C++ Version 7.0 was used as the software development tool for the full program, due mainly to the ease of access to the low level operations resident in the computer operating systems.

8.2.2 Video Blaster Framebuffer

In order to carry out image processing operations on the digitized image, it is necessary to access the Video Blaster Framebuffer, which was used to accommodate the captured raw digital image. This requires a knowledge of the data format and organization of the Framebuffer.

8.2.2.1 Format of the Video Blaster Framebuffer

The Video Blaster uses 768KB memory of video storage. This 768KB is mapped into 1MB of address space that is accessible to the CPU in a linear format.

The format of the Video Blaster framebuffer is a YUV configuration. This format defines Y to be the luminance and UV to be the blue and red chrominance values respectively. These components, which are related to the RGB colour space are as follows:

$$Y = 0.301xR + 0.586xG + 0.113xB$$

$$U = B - Y = -0.301xR - 0.586xG + 0.887xB$$

$$V = R - Y = 0.299xR - 0.586xG - 0.113xB$$

The resolution of Y component, which is also a grayscale value, is an unsigned quantity of 7 bits or a value between 0 and 127. The U and V are signed quantities and are scaled by 127/226 and 127/179 respectively to fit into 7 bits.

This data format has the advantage of obtaining grayscale values without calculation, but also has the inconvenience of extracting color information from

the image. This project mainly deals with grayscale image processing, which means only the Y component was required for processing operations.

The YUV format is encoded as YUV 4:1:1, where there are 4 samples of Y for each sample of U and V. That is , the same U and V value is shared among 4 pixels. The framebuffer data format has the format shown in the following table:

Table 8.2-1 Data Format in Framebuffer

PIXEL#	BYTE ADDR	D15 D8	D7 D0
0	8n+0	U6 U5 V6 V5 X X X X	Y0:6....Y0:0 X
1	8n+2	U4 U3 V4 V3 X X X X	Y1:6....Y1:0 X
2	8n+4	U2 U1 V2 V1 X X X X	Y1:6....Y2:0 X
3	8n+6	U0 X V0 X X X X X	Y1:6....Y3:0 X

The X sign indicates an unused bit. Each pixel is represented as a word (16 bits) in the buffer, where only 11 bits are used. Bits 0 and 8-11 of every word are not used.

8.2.2.2 Organization of the Framebuffer

In the Video Blaster Framebuffer, each scan line begins at the 2KB boundary. As each pixel is presented as a word, this format means that each scan line (2KB) is equivalent to 1K of pixels, which is intended to accommodate the standard video format that consists of about 700 pixels per scan line. Memory mapping of the Frame Buffer data structure is illustrated in Figure 8-3.

From an analysis of the organization of the image data, it can be concluded that the resolution of the video Blaster Framebuffer is 1024×384 pixels. Each Horizontal line consists of 2KB÷2B = 1024 pixels. There are 768KB÷2KB=384 vertical lines.

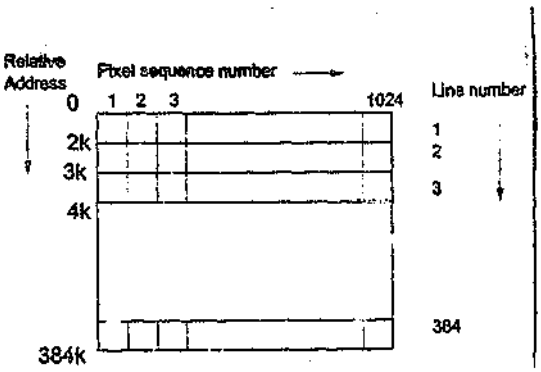


Figure 8-3 Frame Buffer Data Structure

8.2.3 Management of Memory

The image processing algorithms involves processing huge amounts of data. It is crucial to effectively manage the memory space to obtain optimum processing times.

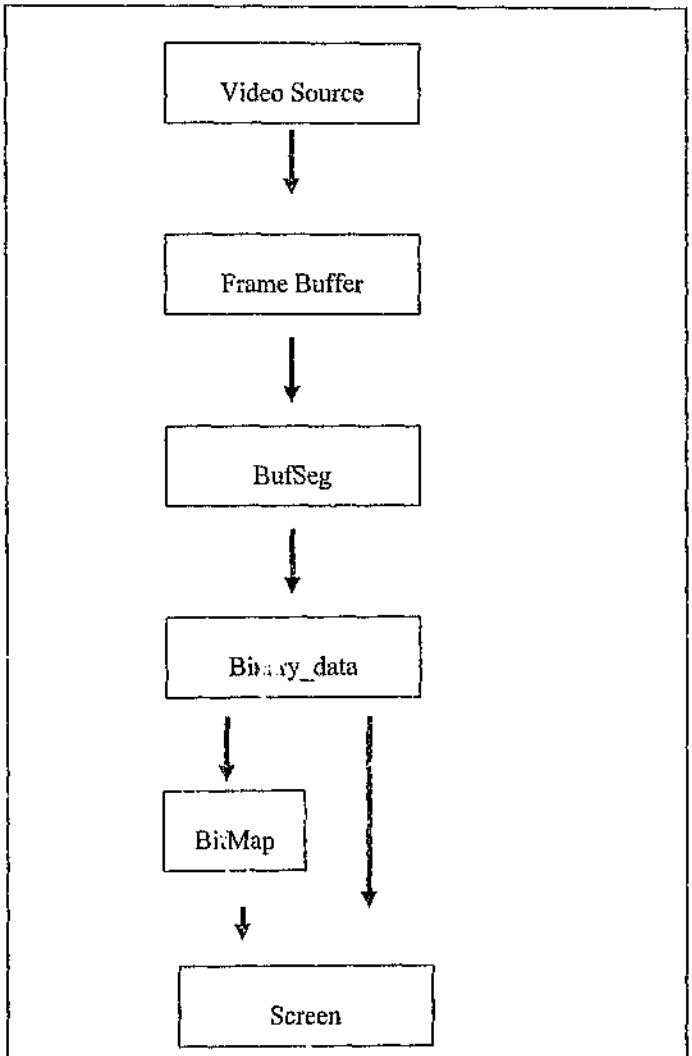


Figure 8-4 Image Data Flow Chart

Figure 8-4 shows the conceptual data flow of the image processing algorithm.

The Video Blaster Framebuffer is allocated 1 megabyte of address space, where only 768 KB is utilised. This is required to be mapped above the system memory

in order to avoid conflicts with other installed software such as Windows which already resides in the system memory above 1MB. This requirement was implemented by appropriate settings of the configuration file. In this project, the Framebuffer address was set as F0. As 16Mb of extended memory is available in the computer, this leaves 15Mb of the installed memory for other programs.

Since the Video Blaster Development Kit does not provide a function for direct access to the Framebuffer, a special interface routine was programmed to read out the grayscale values of the image in the Frame Buffer. The function of this routine was to transfer the image data into RAM, which allows a faster manipulation of the data than in other areas of memory space. The BIOS Interrupt 15H function 87H which provides a service of Block Move between Extended Memory and RAM was used for this transfe.. Because only one segment of data is allowed to be transferred during each inte uption operation, a special segment named BufSeg was designated as the data transition area. In order to transfer the 768Kb of data between the Framebuffer and RAM, 768/64=12 interrupt operations are needed.

The BitMap segment was allocated as an area to accoi modate a frame of image to be displayed on the screen. Preprocessing operations can be carried out in this segment space, this space has same data format as display memory map.

The screen segment was the display memory of the graphics card of the 80486 computer. Under VGA mode 12H x, the segment address for the Screen is A0000.

Coding Scheme: The purposes of carrying out image coding are to enable data compression to be accomplished and to ensure that image processing operations are conducted in the most convenient form.

In the majority of cases during image processing, mathematical operations are performed on the two-dimensional images, either on an entire image or on a window within the image. This requirement means that an image processing buffer has to be provided in which an image is in a normal display format. Many data coding methods such as Runleng encoding can achieve good data compression, however, this is often not convenient for image processing operations, for the reasons stated in Section 4.4.

Initially a bitmapped binary image buffer for the whole Framebuffer was proposed. After thresholding, the image is converted to a binary image, in which each pixel can be represented by a 1 or a 0. This means that each pixel can be represented by one bit in place of one word(16 bits). The 768KB Framebuffer data is reduced16 times into 48K Bytes, which is less than one segment.

Although this approach compressed the image into an affordable memory space, its "one pixel by one bit" coding scheme reduced the image processing operation efficiency due to a series of bit-shift operations. However, this scheme can still be used for preprocessing images to be displayed on screen when real time processing is not required, since it has the same mapping scheme as the screen memory.

In fact only a part of the image in the Framebuffer is of interest and it is only that part that is processed. The "one pixel by one byte" coding scheme is used and is designated as a segment called Binary_data and is in effect an image buffer.

The resolution and space capability of the allocated memory are illustrated as the following table:

Table 8.2-2 Memory Allocation

Segment Name	Resolution (row × column)	Coding Scheme
FrameBuffer	384 × 1024 = 384 KB 200 × 320 = 64000 words (used)	one pixel by one word
Binary_data	200 × 320 =64000 Bytes	one pixel by one byte
BitMap	384 × 1024 = 48 KB	one pixel by one bit
Screen	280 × 640 = 38400 Bytes 200 × 320 =64000 Bytes (used)	one pixel by one bit

8.2.4 8.2.5 Shading Correction and Lighting

Fruit images contain shading effects due primarily to the spherical shape and the dependence of surface reflectance on surface orientation. The shading effects cause distortion in grayscale measurements. Miller (1991) used a linear pixel-by-pixel model to accomplish shading corrections. Correction factors were obtained from digitized images of a diffuse reflectance sphere, in this case a 70mm

diameter foam rubber ball. One disadvantage of this method is that a pixel-by-pixel modification operation is required on the whole image, which involves a considerable amount of time to process an image. Another problem is that large differences exist between the appearance of different fruit samples. Figure 8-5 shows the different shading area of two apples under the same lighting conditions. This implies that modification to the different images may produce poor results in certain cases.

Two shading correction methods were considered: one method eliminated the shading by adjustment of the lighting conditions. In the second method a special

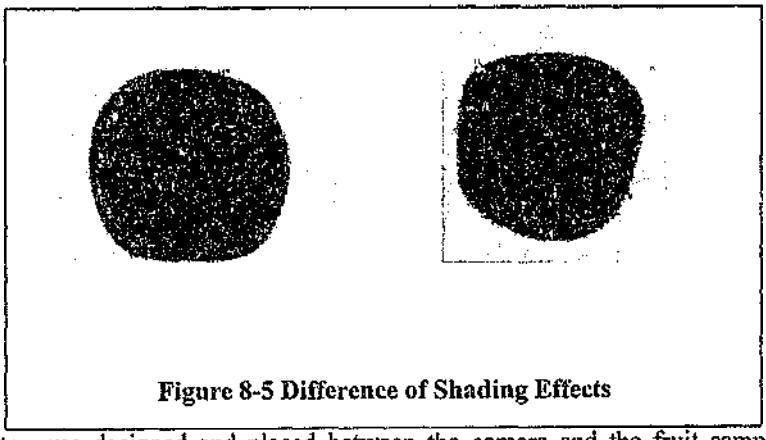


Figure 8-5 Difference of Shading Effects

filter was designed and placed between the camera and the fruit sample to be inspected. In this way, the differing reflectance values of the fruit is catered for.

Obviously the former method is the easiest method to implement. Either a semi-circular light or three lights projected from three directions was found to provide an approximately uniform diffuse illumination on the fruit surface.

A special program was designed to provide the object surface intensity analysis function. Shadows can be avoided by adjusting lighting. Using this program,

proper lighting conditions can be established and set up by carrying out a surface intensity analysis on a sample batch of fruit before analysis of a production batch of fruit inspection is commenced.

8.3 Image Segmentation

8.3.1 Segmentation Method

Edge detection techniques involve operations using different operators such as the Robert gradient, Sobel operator, Prewitt operator, Laplacian operator etc. Picture matching and tracking techniques generally involve a convolution operation. It is apparent that more pixels are involved in the transformation calculation and considerably more time is taken using these methods.

Considering the fact that processing time is crucial for a practical inspection task, the Histogram method of defining the subject and background was chosen as the segmentation method for this project.

8.3.2 Selection of the threshold Value

In general, the gray level histogram of the image is in bimodal form. The valley between the two peaks is used as the threshold value to segment the fruit from the background. Experimental results showed that a manual selection of the threshold value produces good results. Threshold setting using human cognizance has been found by experience to be the most efficient image processing system for the process of segmentation.

An automatic threshold value selection method was also investigated. This method applies a peak searching algorithm to the gray level histogram. A peak is defined as follows:

If the value of point 'X' is bigger than the value of any of the next 'DELT' continuous points, 'X' is called a peak point.

DELT can be decided on the basis of past experience. Experiments carried out on this project showed that the performance is near the optimum if a setting of DELT=10 is selected. If value of DELT is too small, many small peaks will result under conditions of uneven illumination.

An automatic threshold value selection program can be used in any practical machine system in which there is no scope for human intervention. As this involves mathematical operations on the histogram and is sensitive to environmental conditions, longer processing times are needed and poor results may be produced under uneven illumination. For this reason, automatic selection of the threshold was not used in our image processing procedure, however, it has been provided as a separate function if it is required.

For the majority of fruit inspection tasks, the fruits to be inspected are usually viewed under the same lighting conditions, the same location, etc. which means they will have similar histograms and a relatively stable threshold value will result. Based on this consideration, initial tests were conducted on a batch of samples. A series of threshold values were manually chosen by an analysis of the

histograms. The average of this series of threshold values was used as a fixed threshold for subsequent tests on large batches of fruit.

8.3.3 Noise Removal

Even optimum illumination can not eliminate noise, this noise may distort the measurements taken on the image. Figure 8-6 shows how the isolated points affect the measurement of diameter of a sample apple.

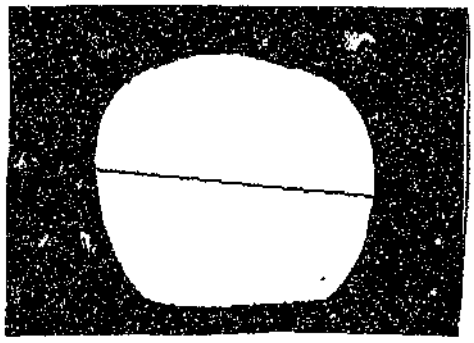


Figure 8-6 Isolated points

Smoothing is an accepted image enhancement technique that is usually applied to remove isolated bright and dark regions of a picture which are caused by signal noise.

One way to smooth an image is to impose the scene by generating multiple picture matrices of the same scene over a given time period. Each matrix will contain identical grayscale picture information, but any random noise will not be represented the same way in any two matrices. A composite matrix is then formed by averaging the corresponding grayscale values of the multiple matrices. This

will cancel out any 'salt-and-pepper' noise due to random variations, without affecting the picture information. This technique is sometimes referred to as ensemble averaging.

Another technique, called local averaging, removes noise by replacing each grayscale value in the image matrix with the average of several values that occur in surrounding pixel windows. This averaging operation, of convolution type, is carried out with the same weighting factor of 1/9. This technique will eliminate noise, but it also washes out and blurs the edges of the image.

In the view of computational complexity, a Boolean Operator operating on the captured image to remove isolated points was used in this program.

The isolated point was defined as a foreground pixel that has no foreground pixels as its neighbours. Based on an 8-neighbour-connectivity definition, a pixel a_0 has eight neighboring pixels:

a_1	a_2	a_3
a_8	a_0	a_4
a_7	a_6	a_5

A local Boolean function to identify pixel a_0 as an isolated point is as follows:

$$B = (\text{NOT } a_1)\text{AND}(\text{NOT } a_2)\text{AND}(\text{NOT } a_3)\text{AND}(\text{NOT } a_4)\text{AND}(\text{NOT } a_5)$$

$$\text{AND}(\text{NOT } a_6)\text{AND}(\text{NOT } a_7)\text{AND}(\text{NOT } a_8)$$

The local Boolean Operator is

$$a_0 \text{ AND } (\text{NOT } B)$$

8.4 Feature Extraction and Measurement

The grading standards serve as an initial reference for the design of the descriptive features for specific fruit. Selection of good descriptive features is crucial for the grading and classification of fruit. A knowledge of the specific kind of fruit and a large amount of statistical data concerning fruit characteristics is required.

In general, the grade standards for fruits specify both the range of size and the extent of defects of individual fruit. The descriptive features are classified into two categories as follows:

1. Geometric Features: features based on contour and shape of the inspected object.
2. Spectral Features - statistical features based on the distribution of the object.

8.4.1 Geometric Features

Geometric features are measured from the binary image.

Diameter

Diameter of the fruit is a good indicator of its size. After thresholding, the captured image is converted into a black and white image. Isolated points are removed using the noise-removal routine. The diameter measurement is then made by identifying the longest, horizontal sequence of low-value, or black pixels.

Area

The object's area was taken by simply counting the number of foreground pixels in the image.

Perimeter

The perimeter was used by counting the number of pixels in the P_8 and P_4 perimeters. The product is a good approximation to P^2. Details on how to compute P_4 and P_8 are given in Section 4.5.2.1.

Shape Indices

Shape indices are used to discriminate geometric shape. The generalized pi index - P^2/Area ,which is independent of changes in scale and rotation and is easy to compute, is regarded as a good index of the simple geometric shapes such as squares, triangles and ellipses. But it does not have very good discrimination powers for irregular shapes.

The reverse of pi is called compactness ratio, which attains its maximum when the figure is a circle. For a regular polygon, it is defined as:

$$Compactness\_ratio = \left(\frac{\pi}{m}\right) \times \cot\left(\frac{\pi}{m}\right)$$

which increases monotonously to 1 as m increases.

For a general shape, there is a theoretical solution to the shape discrimination problem, this is based on calculating the moments of the object. Even though moments can be used to classify objects based on shape, calculating moments is a very time-consuming task.

On average, a batch of fruit will have similar shapes, so for this project it was concluded that the calculation of shape indices was not necessary. However, a routine to carry out this analysis is included in the coding for the analysis of samples.

8.4.2 Spectral Features

Spectral features indicate the fruit surface intensity variation. The variation of these features was used to identify defective areas.

Mean Surface Grayscale

Mean surface grayscale of the inspected object is an important statistical parameter indicating the distribution of surface intensity. When a threshold value T is selected to separate the object from its background, the Mean Surface Grayscale can be calculated as follows:

$$G = \frac{\sum_{i=T}^{128} i \times pixel\_number[i]}{Area}$$

where pixel_number(i) is referred to as number of pixels with grayscale value i.

Defective Ratios

Three defective ratios were proposed in the project. After selecting three cut-off points on the surface gray level histogram, four areas were derived from calculating the number of pixels falling into the following ranges:

Range 1 : Threshold value T to cut-off point 1

Ratio 1 = A1 / A

Range 2 : Threshold value T to cut-off point 2

Ratio 2 = A2 / A

Range 3 : Threshold value T to cut-off point 3

Ratio 3 = A3 / A

Range 4 : cut-off point 1 to cut-off point 3

Ratio 4 = A4 / A

The establishment of the above ratios is made on the basis of the assumption that the reflectance of the defective area is lower than the mean surface grayscale. The reflectance of the defective area is usually a fixed percentage below the value of the mean surface grayscale.

Cut-off points were chosen on the basis of practical experiments and based on the Mean Grayscale G. For grading green apples these points were established as follows: cut-off point 1 was chosen as 8%°G, cut-off point 2 as 5%°G, and cut-off point 3 as 2%°G. (See Figure 8-7).

Experiments conducted on the green apples showed that Ratio 3 is a good indicator for a brown area. It was also observed that the use of a green band pass filter increased the visual differences in the green apple image.

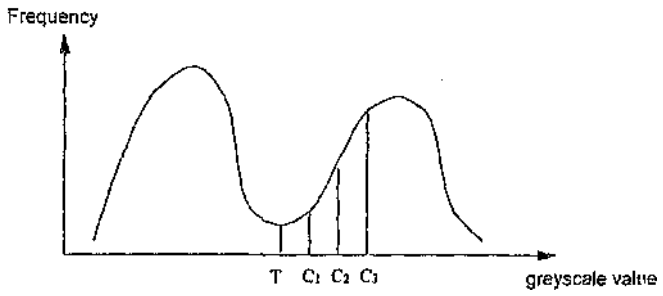


Figure 8-7 Cut-off Points

8.5 Design of the Fuzzy Classifier

The Design of a good classifier is an important topic in pattern recognition studies. In this section, the techniques commonly used in the engineering applications are discussed, followed by an introduction to the concept of fuzzy sets and their application in pattern classification. Finally, details of the design of the fuzzy classification mode and the construction of the feature membership function will be explained.

8.5.1 Selection of Pattern Classification Techniques

In the view of engineering aspects of pattern classification, pattern classification techniques can be classified as Template Matching and Feature extraction.

Template Matching refers to the comparison of the test pattern with a number of stored patterns until an exact match is found. Feature extraction starts from the test pattern and measures a limited number of features that are known, in advance, to be good descriptors for the pattern.

Template matching is both computation and memory intensive when all but the simplest patterns are considered. In addition, it is sensitive to the exact location of the object, unless autocorrelation techniques are used. However, this kind of computation can be performed quickly using optical techniques. Collings (1988) designed an optical pattern recognition system with an optical correlator to perform template matching, using holographic techniques.

Feature Extraction techniques require less expenditure, both in computation and in memory. However, the fundamental problem is that important information may be lost due to extensive data reduction at the pre-processing data stage. The performance of the classification is limited by the quality of the feature measurements provided. In addition, the identification task requires greater computing power. Some specialized computer architectures have been developed to enable higher throughput, however the limitations imposed by the hardware means that the process is becoming more task-dependent and this results in higher costs.

If it is considered that cost-effectiveness and processing time are important factors for a machine vision system in practical inspection applications, then a classifier

designed using feature extraction techniques combined with fuzzy set concepts should prove to be an elegant and optimal solution.

8.5.2 Fuzzy Sets

8.5.2.1 The Fuzzy Sets Concept

The concept of fuzzy sets first arose from the study of problems related to pattern classification(Zadeh, 1965). This is not surprising since pattern recognition is an important aspect of human perception, which use fuzzy processes naturally. Pattern recognition problems still pose serious challenges to machines even though they have been studied over a number of decades. However, these problems can be easily solved by humans. A human can easily recognize an object in diverse lighting conditions, even if the scene is occluded or obscured by a film of dirt. The perception of the scene results in an infinite number of 3D objects being projected on to a two-dimensional retinal image, however, the brain interprets these images with little difficulty, often filling in the missing dimension. One difference between human perception and that of a machine, is that humans perform their reasoning using fuzzy logic rather than two valued logic. Fuzzy logic differs from conventional logic systems in that it aims at providing a model for approximation rather than one for precise reasoning.

Let U be a Universe of discourse, a fuzzy subset A of U (Zadeh,1965) is defined by a membership function $f_A : U \rightarrow [0,1]$, which associates with each element u of U a number $f_A(u)$ in the interval [0,1], where $f_A(u)$ is defined as the grade of membership of u in A. Formally, A is written as

$$A = \{ (u, f_A(\mu)), u \in U \}$$

from the above definition, it can be seen that fuzzy sets allow a gradual rather than an abrupt transition, from membership to nonmembership.

Bellman and Zadeh (1966) extended the application of fuzzy sets into pattern classification in their pioneering paper as follows:

Let A and B denote two fuzzy sets in U, with f_A and f_B denoting their respective membership functions (also called characteristic functions). Suppose that we are given n ...mple triplets $(x^1, f^1_A, f^1_B),...,(x^n, f^n_A, f^n_B)$, with (x^i, f^i_A, f^i_B) representing a sample consisting of x^i and the values of f_A and f_B at x^i. The problem of pattern classification in this context is essentially that of estimating the membership functions f_A and f_B from the given features instead of the features themselves.

8.5.2.2 Fuzzy Classification Models

There are two general approaches to solve pattern classification problems with fuzzy sets: either using semantic pattern recognition or syntactic pattern recognition.

The idea behind syntactic pattern recognition is that certain pattern classes contain objects that have an identifiable hierarchical structure that can be described by a formal grammar, called the pattern grammar. A basic set of pattern primitives is selected and forms the set of terminals of the grammar. This results in a grammar which is a list of allowable relations among the primitives. The pattern is a set of

strings generated by the pattern grammar. Lee (1973) discussed the possibility of applying fuzzy grammars to the recognition of leukocytes and chromosomes.

Semantic pattern recognition represents a fuzzy pattern class in a feature space (Kotoh and Hiramatsu,1973). A feature is viewed as a fuzzy pattern space, i.e., each member of the fuzzy partition corresponds to a fuzzy value of this feature. A fuzzy pattern class is expressed by a logical expression of feature values that correspond to different features:

Let F be a fuzzy pattern class defined by the fuzzy feature values F_1 , F_2 ,... F_i ,..., F_r , where Fi is a fuzzy value of feature i. An object p is thus characterized with respect to the class F by r membership values $\mu_{Fi}(m_i(p))$ denoted $\mu_i(p)$ for convenience. m_i is the measurement procedure associated with feature i and $m_i(p)$ is the feature value. $\mu_F(p)$ which is then constructed by aggregating the values of $\mu_i(p)$ in some manner. The choice of an aggregation scheme is problem-dependent.

8.5.2.3 Application of Fuzzy sets

The Fuzzy set concept is now well developed and has found applications in a wide range of scientific areas, such as Artificial Intelligence, Robotics, Medical Diagnosis, System Control, Transportation Planning, Economic Modeling, etc.

In the field of industrial engineering, Fuzzy sets have been generally employed in the field of process control, a feature of which are approximate and uncertain inputs. Billitos (1993) used fuzzy pattern recognition techniques for shape control in the Ballscrew grinding process. Pham and Hafeez (1992) designed a fuzzy qualitative model for a robot sensor to locate three dimensional objects by using

fuzzy relation matrices. But the on-line use of this technique was found to be impractical due to the heavy computational burden.

The attraction of fuzzy classification lies not only in its intrinsic use of the vagueness of an observed pattern or input noise caused by the working environment, but also in its evaluation procedure based on a fuzzy set theoretic method of inference. The simplicity of this evaluation method, which only consists of minimum and maximum operators, also results in the advantage of a reduction in computing time.

Fuzzy sets tend to be applied in an environment in which humans have traditionally intervened. In the case of fruit inspection, a human would judge the "goodness" of an inspected item without a sharp boundary. Using the concept of "degree of goodness" in the fruit inspection process, is a novel idea, which introduces human perception into fruit grading and inspection processes.

8.5.3 Classifier Design

The fuzzy classifier in the project, was based on Semantic Pattern Recognition, proposed by Kotoh and Hiramatsu(1973). Each object is evaluated with respect to a fuzzy pattern class by means of a fuzzy logical expression of features, as indicated in Table 8.5-1:

Table 8.5-1 Fuzzy Logical Expression of Features

	Diameter	Blemish
Class 1	large	small
Class 2	medium	small

If class 1 is denoted by fuzzy set A, and class 2 by fuzzy set B, a fruit being inspected can be described by a feature vector x, as follows:

$$x = \begin{bmatrix} d \\ r \end{bmatrix}$$

where d = diameter,

r = defective ratio.

Using the concept of membership functions, fuzzy set A (Class 1) can be expressed in terms of, $\mu_A(d)$ and $\mu_A(r)$, while fuzzy set B (Class 2) can be expressed in terms of, $\mu_B(d)$ and $\mu_B(r)$,

$$\mu_A = MIN[\mu_A(d), \mu_A(r)]$$
$$\mu_B = MIN[\mu_B(d), \mu_B(r)]$$

Decision Rule:

By setting a membership function threshold, δ, where $0 \leq \delta \leq 1$, and comparing $MAX[\mu_A, \mu_B]$ with δ, two possibilities emerge (Lee,1975):

i) If $MAX[\mu_A, \mu_B] > \delta$, then:

if the maximum is unique, the class corresponding to the maximum value is chosen and the image is classified accordingly.

if the maximum is not unique, relative priorities are assigned to μ_A and μ_B,, and the image is classified accordingly. In the experiments carried out, MAX[$\mu_A(d),\mu_B(d)$] has been designated as the priority for the grading of green apples.

ii) If MAX[μ_A, μ_B] $\leq \delta$, then the image is rejected as not belonging to either of the two classes.

8.5.3.1 Construction of Membership Functions

The membership function is purported to be a good model for the way people perceive categories. However the methods of deriving membership functions lack generality.

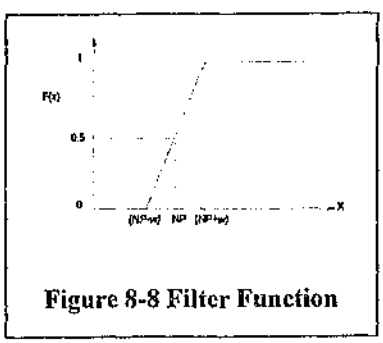


Figure 8-8 Filter Function

In this project the fea.. membership function has been built on the basis of a specific filter function (see Figure 8-8), proposed by MacVicar-Whelan(1978):

$$F(x;NP,w)=\begin{cases} 0 & (\text{if } x \in [-\infty, NP-w]) \\ \dfrac{(x-NP+w)}{2w} & (\text{if } x \in [NP-w, NP+w]) \\ 1 & (\text{if } x \in [NP+w, +\infty]) \end{cases}$$

where: NP = neutral point,

$2w$ = width of the transition between membership and non-membership.

Values of the membership function were obtained from statistical analysis of a representative fruit sample. For each feature, a normal probability distribution was fitted to the frequency histogram of observed values. The resulting curve was then used to define the values associated with each class of membership. In the case of the diameter feature, "large" was defined by the membership function:

$$\mu_1(x) = F(x;\bar{x}+\alpha\sigma;\beta\sigma)$$

where $\bar{x}$ = mean value

σ = standard deviation

α, β = user-defined parameters

α and β are determined experimentally, details of which are explained in the following section.

8.5.3.2 Selection of Parameters

The parameters α and β in the membership function

$$u_1(x) = F(x;\ \bar{x}+\alpha\sigma;\beta\sigma)$$

are set by analyzing the statistical distribution curve of a feature.

Suppose the data of the measured features of the fruit samples have the normal distribution, by definition, standard deviation units measured off along the base line, starting from the mean, always cut off certain proportions of the area under

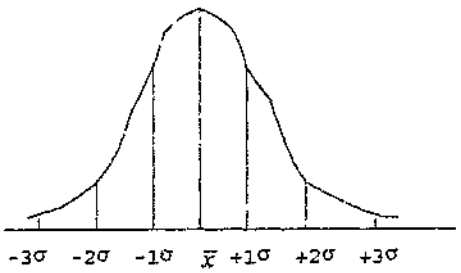


Figure 8-9 Normal Probability Distribution

the curve (see Figure 8-9).

Figure 8-9 shows the area under the normal curve at various standard deviation units from the mean. If one standard deviation units on each side of the mean $\bar{x}$, is measured off, approximately 68 percent of the area under the curve is included. If two standard deviation units on each side of the mean is measured off, approximately 95 percent of the area under the curve is included between these two points. And for three standard deviation units, over 99 percent of the area under the curve is included.

On the basis of the above discussion, a neutral point NP is chosen and a width 2w of the transition in the filter function is defined according to the category being considered.

For the Class 1(Grade Large):

Suppose the descriptive feature diameter, d, has the mean value $\bar{d}_L$,and standard deviation σ_L, the transition range was chosen as $[\bar{d}_L - 3\sigma_L, \bar{d}_L]$, see Figure 8-10, therefore the design parameters will be.

$\alpha = -1.5$; and $\beta = 1.5$

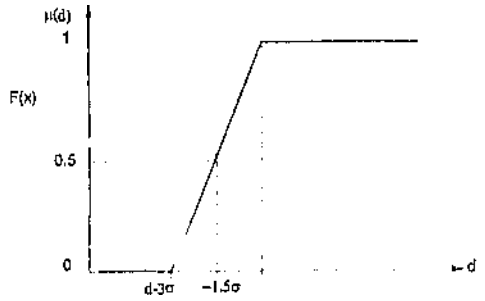


Figure 8-10 Membership Function of Diameter

The membership function for the diameter d is derived as

$$\mu_L(d) = F(d; \bar{d}_L - 1.5\sigma_L; 1.5\sigma_L) = \begin{cases} 0 & \text{if } d < \bar{d}_L - 3\sigma_L \\ (d - \bar{d}_L + 2\sigma_L)/2\sigma_L & \text{if } d \in [\bar{d}_L - 3\sigma_L, \bar{d}_L] \\ 1 & \text{if } d > \bar{d}_L \end{cases}$$

For the defective ratio r, the transition range was selected as $[\bar{r}_L, \bar{r}_L + 2\sigma_{rL}]$, (see Figure 8-11), with

$$NP = \bar{r}_L + \sigma_{rL}$$

$$w = \sigma_{rL}$$

where $\alpha = 1, \beta = 1$.

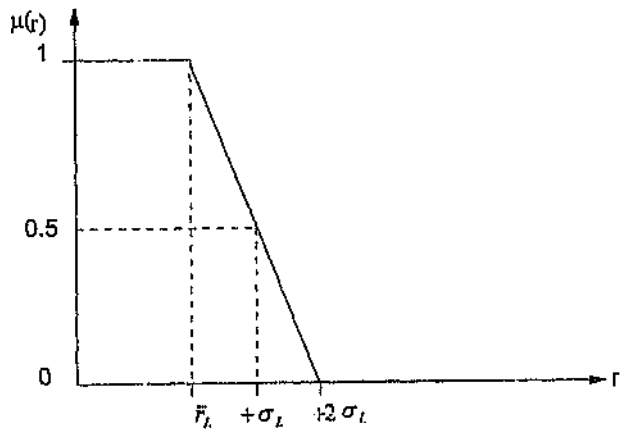


Figure 8-11 Membership Function of Defective Feature

The membership function of feature r was derived as

$$\mu_L(r) = F(r; \bar{r} + \sigma_{rL}; \sigma_{rL}) = \begin{cases} & r > \bar{r} + 2\sigma_{rL} \\ (\bar{r} + \sigma_{rL} - r)/\sigma_{rL} & r \in [\bar{r}, \bar{r} + 2\sigma_{rL}] \\ 1 & r < \bar{r} \end{cases}$$

For the Class 2(Grade Medium):

Suppose the descriptive feature diameter d has the mean value $\bar{d}_M$, and standard deviation σ_L, two transition ranges were chosen: $[\bar{d}_M - 3\sigma_M, \bar{d}_M]$ and $[\bar{d}_M + \sigma_M, \bar{d}_M + \sigma_M]$, see Figure 8-12.

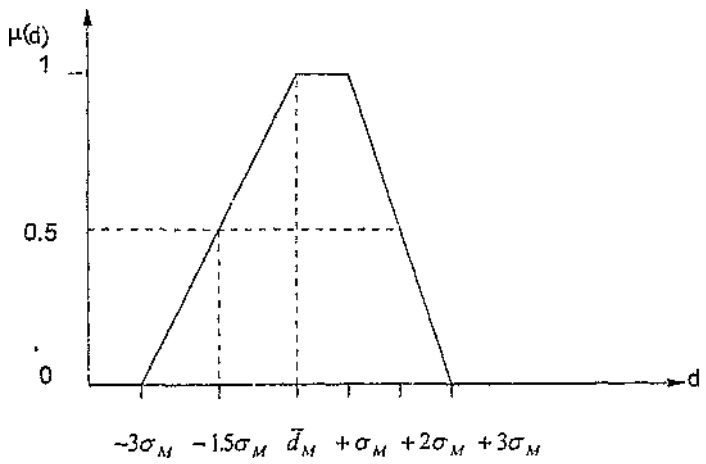


Figure 8-12 Membership Function of Defective Feature

The membership function for the diameter d is derived as

$$\mu_L(d) = \begin{cases} 0 & \text{if } d < \bar{d}_M - 3\sigma_M \\ (d - \bar{d}_M + 2\sigma_M)/2\sigma_M & \text{if } d \in [\bar{d}_M - 3\sigma_M, \bar{d}_M] \\ 1 & \text{if } d \in [\bar{d}_M, \bar{d}_M + \sigma_M] \\ (\bar{d}_M - d + 3\sigma_M)/2\sigma_M & \text{if } d \in [\bar{d}_M + \sigma_M, \bar{d}_M + 3\sigma_M] \\ 0 & \text{if } d > \bar{d}_M + 3\sigma_M \end{cases}$$

For the defective ratio r, the transition range was selected as similar to the range in the Class 1.

The membership function of feature r was derived as

$$\mu_M(r) = F(r;\bar{r}+\sigma_{rM};\sigma_{rM}) = \begin{cases} 0 & r > \bar{r}+2\sigma_{rM} \\ (\bar{r}+\sigma_{rM}-r)/\sigma_{rM} & r \in [\bar{r},\bar{r}+2\sigma_{rM}] \\ 1 & r < \bar{r} \end{cases}$$

8.6 Construction of the Template Model

The general requirement is to establish a data model; this is a template, the object of which is to accomplish the object recognition and classification tasks in a machine vision system. In order to define the data model it is necessary to explore the solution to the problem. This is the major task to be resolved in the application of learning to machine vision systems.

The study of learning techniques for machine vision systems stems from a strong desire by researchers for computers and computer controlled robots to be able to perform a task using self-teaching techniques. At present, the field of machine learning is organized around three primary research foci:

1) Task-Oriented Studies - the development and analysis of learning systems to improve performance in a predetermined set of tasks.

2) Cognitive simulation - the investigation and computer simulation of human learning processes.

3) Theoretical Analysis - the theoretical exploration of the space of possible learning methods and algorithms independent of application domain.

The first topic is of great interest for industrial engineering, in which the learning systems are task - dependent.

Learning strategies for machine learning can be broadly classified as follows:

1) Direct implanting of new knowledge - such as learning by being programmed.

2) Learning from instruction.

3) Learning by analogy.

4) Learning from examples.

5) Learning from observation and discovery.

Learning from examples is a method that has been heavily investigated in Artificial Intelligence. Given a set of examples and counter examples of a concept, the learner induces a general concept description that describes all of the positive examples and none of the counter examples. Most machine learning system are based on this method.

In machine vision systems, the goal is to obtain, through learning, descriptions of local features associated with the object. These features are then processed by specific techniques and used to construct a data model, which is then a template of the object, for the recognition task.

8...MACHINE VISION AND FUZZY SETS IN FRUIT CLASSIFICATION

Current machine vision systems usually acquire a "model" of an object by one of four approaches:

(1) Have the system acquire the model by showing it an example (and then analyzing the image of that example).

(2) Provide a computer-readable "drawing" of the object (the CAD approach).

(3) Embed a description in a special-purpose program.

(4) Model the object in a simple, semi-natural language description.

The other machine vision system described in this thesis uses knowledge-based modeling techniques intended for accomplishing complex automated industrial tasks. But for most machine vision systems, the traditional approach of acquiring a model is still " Learning by Showing ", in which a user shows the system several examples of an object, and the system gathers statistics that are used to estimate the expected values of the object's features and their variances. Jamshidi (1990) while working at the Stanford Research Institute used similar methods to develop an image processing algorithm which was quite successful.

For the task considered here, which is the analysis of a relatively simple scene, the " Learning by Showing " approach was chosen to acquire the template model for the fruit to be inspected.

The function of constructing a template model, is performed by selecting samples of different categories of fruit. For each grade category, the samples are viewed in

front of the camera. Measurements are taken of the descriptive features. After testing the whole batch of samples of the same category is completed, statistical calculations for each of the descriptive features are carried out. The mean value and standard deviation of each descriptive feature are obtained and the results are compiled in a data file which is loaded into the template model when the system switched to classification mode.

8.7 Implementation

To implement the basic tasks of this machine vision system, five basic modules are required: the Image Acquisition, Image Analysis, Learning, Classification, and I/O interface modules. A description of each of these modules is given in the following sections.

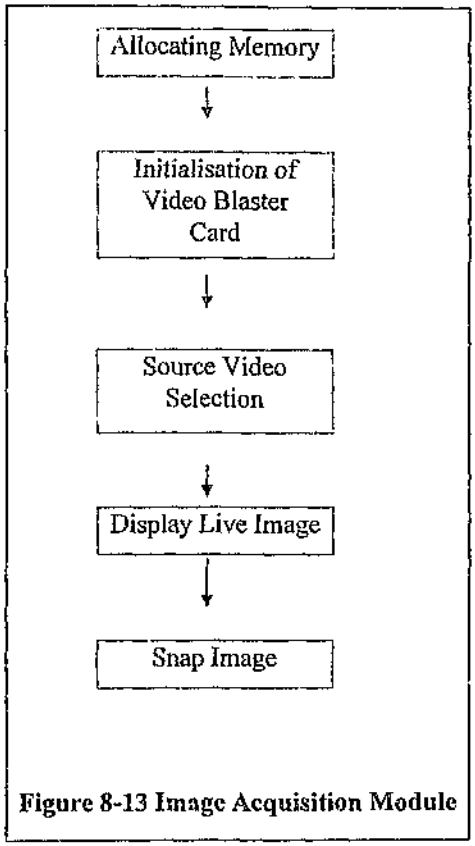


Figure 8-13 Image Acquisition Module

8.7.1 Image Acquisition Module

The Image Acquisition Program starts by allocating memory to three data buffer areas designated as BufSeg, Binary_data and BitMap. The Video Blaster Grabber Card is initialized by checking for the existence of the resident driver VBLSTDRV. By calling this driver with the appropriate interrupt vector number, a sequence of functions interfacing with the Grabber are performed. A live image from the camera is displayed through a specific window. The image is snapped by freezing the Framebuffer. This sequence of operations is shown in Figure 8-13.

8.7.2 Image Analysis Module

The Image Analysis module consists of four basic routines: **Histogram Computation, Thresholding, Surface Grayscale Analysis** and **Feature Analysis.**

The **Histogram Computation** routine computes the frequency distributions of the pixel-intensity values on the snapped image in the Frame Buffer. The results in the form of a graph are displayed on the monitor screen. Hard copy can be obtained using an on-line printer.

The **Thresholding** routine performs the function of extracting the object of interest from the background. It can be switched into one of three operational modes: Manual Thresholding, Auto thresholding, and Default Thresholding.

In manual thresholding mode, a histogram of the snapped image is displayed on the screen. After a threshold value is set by the user, a binarized image is put into the Binary_data segment and displayed on the screen. A good threshold value can be found by comparing the binary images. This mode is the default mode during the learning phase.

As an alternative, auto thresholding can be used to perform the histogram analysis function. The appropriate threshold value can be found using a peak searching procedure. At the present stage of system development, this method is purely used to compare the results obtained by this method with those of manual thresholding. In the future, depending on results, this method may be used as the default method.

In the default thresholding mode, the threshoid value is taken from the template data file.

The **Surface Grayscale Analysis** routine performs the surface analysis function by providing a pseudocolour image on screen. The surface intensity of the object is divided into four ranges, with one of the four colours blue, light blue, green or red being assigned. The four ranges are set inter-actively by the user. The user can observe the intensity changes in a proactive manner. When the system is in configuration mode, this routine is used to adjust the lighting conditions.

Pseudocolouring is a technique that is used in computer graphics, the technique is used to replace a range of gray pixel values with colour in the subject image. In this way it is possible to identify the image content according to a correlation between pixel values and the informational content of an image.

The **Feature Analysis** routine takes measurements on the descriptive features of the object. This operation is carried out on the Binary_data segment.

8.7.3 Classification Module

Classification begins by loading the sample template into the template model by reading the template data file. Based on this template, the membership functions of the descriptive features are established.

The second step of the process is to make a decision. Descriptive features are sent to the fuzzy classifier and their membership function values are computed. According to the decision rule described in Section 8.5.3, the fruit is classified into specific categories. The classification results are displayed on the screen.

8.7.4 Learning Module

The basic function of learning module is to perform statistical calculations on the descriptive features of a series of sample fruits. The results are:

1) mean diameter and its st. dard deviation;

2) mean defective ratio 3 and its standard deviation;

3) mean defective ratio 4 and its standard deviation;

4). classification threshold δ

5) a fixed threshold value for image segmentation.

The results are output to a template data file which will be used to construct the template model when system is in classification mode.

8.7.5 I/O Interface

The I/O interface module performs the following three functions:

- Interface with User
- Interface with data files
- Interface with printer

Interfacing with User is implemented by a command Menu and an input handler. Control commands can be input via the keyboard or by a mouse.

Interfacing with data files is implemented by a routine with **Read File**, **Write File** and **Template file** options.

Images and information displayed on the screen can be printed by the Shift-Print-Screen function key under graphics mode.

8.8 Experimental Results and Discussion

8.8.1 Experimental Results

Experiments have been conducted on the fresh green apples of varying degrees of acceptability.

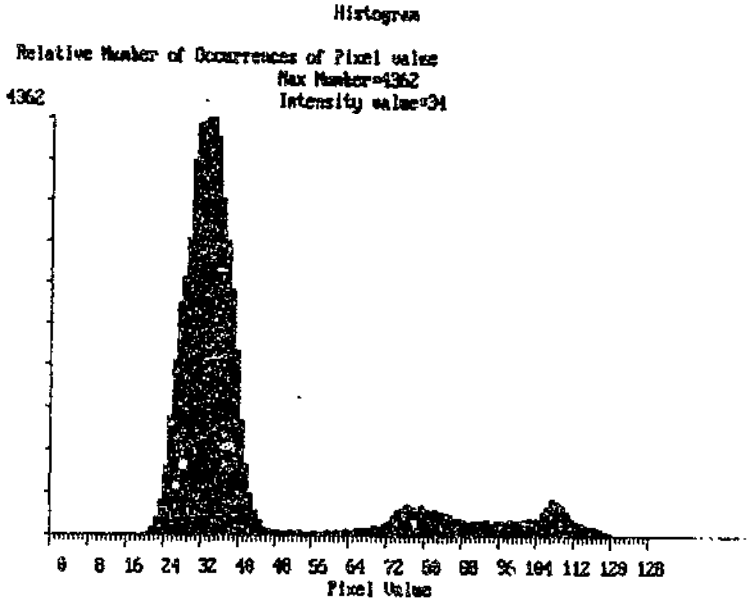


Figure 8-14 Without Color Filter

First, the system was set in the configuration mode. The lighting condition was set up by running the surface grayscale analysis function. Figure 8-14 shows the pixel intensity histogram for an image captured without a green color filter. A pixel

intensity value of 0 indicates black, while a pixel intensity value of 128, indicates white. The high frequency of dark pixels in the range 20 to 48, corresponds to the dark background against which the image was captured. Pixel values to the right of this region indicate the actual fruit specimen.

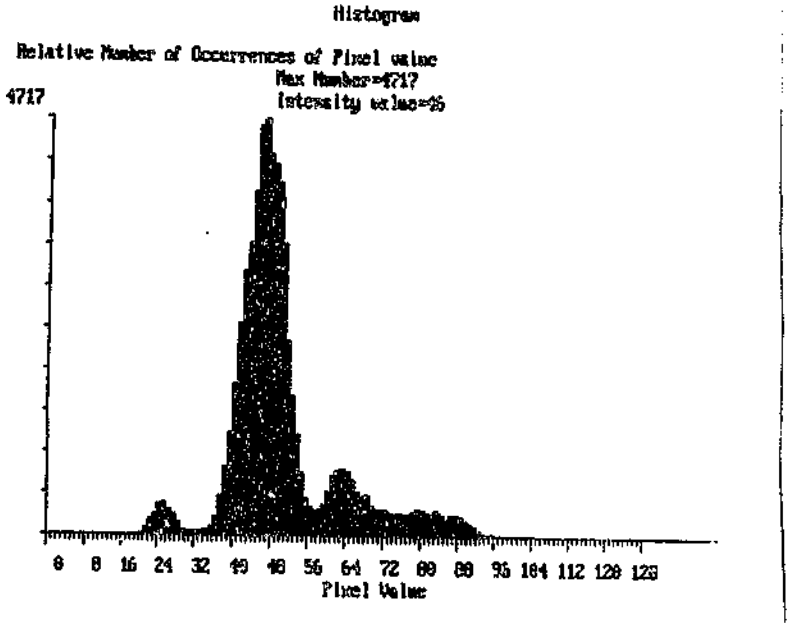


Figure 8-15 With Color Filter

The effect of using the color filter on the same specimen, is illustrated in Figure 8-15. In this particular case, the specimen had a blemish which became apparent when the color filter was applied. The blemish corresponds to the high-frequency peaks in the pixel intensity rang, 60 to 66.

Next, the system was switched to learning mode. Samples were taken from two grades of apples. Table 8.8-1 is an extract from the template data file for a grade 1 green apple. This data was compiled during the learning phase.

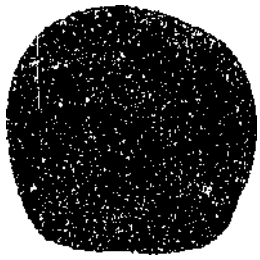
Table 8.8-1 Template Data

```
Number of samples: 10
D  mean  area   P      PI     ratio1  ratio2  ratio3  ratio4
175  88  32408  1000   0.400  0.147   0.233   0.327   0.180
180  87  31270  1000   0.393  0.113   0.150   0.235   0.121
174  87  31125  1007   0.365  0.119   0.162   0.267   0.148
165  86  33165  960    0.445  0.128   0.182   0.318   0.190
167  89  33489  958    0.458  0.122   0.241   0.372   0.250
176  88  31006  1000   0.390  0.127   0.192   0.307   0.179
183  87  32617  1008   0.346  0.162   0.203   0.280   0.118
164  89  32148  1063   0.357  0.091   0.170   0.325   0.234
173  90  33107  1014   0.404  0.178   0.245   0.333   0.155
171  90  33019  997    0.417  0.177   0.237   0.316   0.140
162  90  34845  1155   0.328  0.174   0.219   0.282   0.108
170  88  32341  976    0.426  0.113   0.180   0.276   0.163
177  88  31469  988    0.405  0.109   0.173   0.291   0.102
178  89  32607  991    0.409  0.154   0.240   0.343   0.189
166  87  32161  990    0.412  0.116   0.163   0.300   0.184
183  87  30414  1034   0.357  0.133   0.179   0.297   0.164
167  86  32859  1057   0.370  0.130   0.173   0.272   0.142
172  88  32575  970    0.435  0.082   0.207   0.354   0.272
 Mean
172  88  32339  1014   0.396  0.132   0.197   0.305   0.173
 Standard deviation
6.33 1.25 161.19 47.83 0.0339  0.0273  0.0307  0.0333  0.0431

 threshold value: 64
 Cut point1=78 Cut point2=80 Cut point3=83
```

Following the learning phase, the system was switched to classification mode. The results from the classification are as follows:

Figure 8-16 is an acceptable apple image displayed by the **Grayscale Analysis** Function in the **Analysis** submenu. No apparent blemishes were observed on the apple surface.



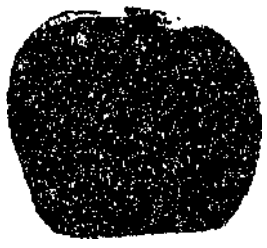
```
Grayscale distribution analysis by assigning color to selected ranges
Range0: [0--threshold] (black color)point1= 00
Range1: [threshod--point1] (blue color)point1= 60 ratio1=0.143994
Range2: [point1—point2]   (Green color) point2= 83 ratio2=0.188860
Range3: [point2—point3](light blue color) point3= 86 ratio3=0.247148
Range4: >point3 (  Red color       )
point1 -- point3 ratio=0.103154
```

```
mean_d= 172 std_dev_diameter= 6.330
mean_ratio3=0.305000 std_dev_ratio3= 0.033300
mean_ratio4=0.173000 std_dev_ratio4=0.043100
Threshold= 64 Delt=0.400000
Diameter= 174 Ratio4=0.103154 Ratio3=0.247148
membership function u(d)= 1.0
membership function u(r)= 1.0
threshold delt= 0.4
It belongs to this class
```

Figure 8-16 An Acceptable Apple

The final stage in the process is the classification phase. For the sample shown in Figure 8.16, the membership function values for diameter and defective were both 1. Under these circumstances, there is no need for comparison with another class, the conclusion drawn is that the fruit is graded as grade 1.

Figure 8-17 is the surface analysis image of a rejected apple. The blemish area in the center of the apple surface is apparent.



```
Grayscale distribution analysis by assigning color to selected ranges
Range0: [0--threshold] (black color)point1= 80
Range1: [threshod--point1] (blue color)point1= 80 ratio1=0.152719
Range2: [point1--point2]   (Green color) point2= 83 ratio2=0.235537
Range3: [point2--point3](light blue color) point3= 86 ratio3=0.334542
Range4: >point3 (  Red color        )
point1 -- point3 ratio=0.181823
```

Figure 8-17 A Rejected Apple

The classification result shows that the defective ratio membership function value is 0, and the diameter membership function value is 1. In this case, the program simply rejects this specimen without further comparison.

8.8.2 Discussion

The fuzzy classifier is currently being refined, but initial results have proved the feasibility of using image processing and fuzzy logic for fruit classification. By using a standard, computer-based system, the concept of automated inspection will be economically feasible for individual farmers in the fruit export industry. Although the concept has yet to be tested under field conditions, it is predicted that a throughput rate of at least 2 items/second will be possible with a relatively low-cost system. The primary limitation of the system is that special lighting is crucial for [illegible] scale measurements

REFERENCES AND BIBLIOGRAPHY

Agapakis, John E. Masubuchi, Koichi (1985), "remotely manipulated and autonomous robotic welding fabrication in space", Proceedings of SPIE - The International Society for Optical Engineering Space Station Automation. v 580 1985 Cambridge, MA, Engl p 68-77.

Anderson,J. M. and Bezdek, J. C., (1984), Curvatures 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. PAMI, vol. 6, pp27-40.

Ahmed, N. and Rao, K. R., (1975), Orthogonal Transforms for Digital Signal Processing, Springer-verlag, Berlin and New York.

Archibald, C., and E.petriu, Eds. (1992), Advances in Machine Vision 1992 World Scientific Publishing Co. Pte. Ltd. ISBN 981-02-0976-2.

Attneave, F., (1954), Some information aspects of visual perception, Psychol. Rev., Vol. 61, no. 3, pp183-193.

Bellman, R., R. Kalaba and L. Zadeh, (1966), Abstraction and pattern classification, J. Math. Anal. Appl. Vol. 13, pp1-7.

Bezdek, J. C., (1981), Pattern recognition with fuzzy objective functions algorithms, Plenum Press,ISBN 0-306-40671-3.

Billatos, S.B.,(1993) Knowledge-based expert system for ballscrew grinding, Journal of Engineering for Industry, Vol. 115 pp230-235.

REFERENCES AND BIBLIOGRAPHY

Bush, V., 1945, "As We May Think," Atlantic Monthly 1976, No. 1 July, pp101-108.

Bulley, N.R., MacDonald, N. and Sziklai, O., (1984), Utilizing image analysis for determining seed viability, ASAE Paper No. 84-501, St. Joseph, MI:ASAE

Castleman, K. R., (1979),Digital Image Processing, Prentice-Hall, New Jersey.

Chen, P.S., (1989), A data-driven intermediate level feature extraction, IEEE Trans.PAMI, Vol. 11, No. 7.

Cheng, F., Hsu, W. and Chen,M., (1989), Recognition of Handwritten Chinese Characters by Modified Hough Transform Techniques, IEEE Trans. PAMI, Vol. 11, No. 4, pp429-439.

Cheng, J. and H. Don, (1991), Segmentation of bilevel images: a morphological approach, in Pattern Recognition: Architecture, Algorithm & Application, World Scientific Co., Eds by R. Plamondon &H.D. Cheng ,pp141-186

Collings, N., (1988), Optical Pattern Recognition Using Holographic Techniques, Addison-Weley , ISBN 0-21-14549-9

Cooley, J. W. and Tukey, J.W., (1965), An algorithm for the machine computation of complex Fourier series, Math. Computat. 19(April) pp-297-301.

Churchill, D.B., Bilsland, D.M., and Cooper, T.M., (1992), Comparison of machine vision with human measurement of seed dimensions, Transactions of the ASAE, vol. 35(1), pp61-64.

REFERENCES AND BIBLIOGRAPHY

Churchill, D.B., Bilsland, D.M., and Cooper, T.M., (1993), Seperation of mixed lots of tall fescue and ryegrass seed using machine vision, Transactions of the ASAE, vol. 36(5), pp1383-1386.

Das, K. and Evans, M.D., (1992), Detecting fertility of hatching eggs using machine vision II: neural networking classifiers, Transactions of the ASAE, vol.35(6), pp2035-2041

Das, K. and Evans, M.D., (1992), Detecting fertility of hatching eggs using machine vision I. histogram characterization method, Transactions of the ASAE, vol.35(6), pp1335-1341.

Davis, Larry S. Kushner, Todd R.; Le Moigne, Jacqueline J.; Waxman, Allen M. (1986), "road boundary detection for autonomous vehicle navigation", Opt Eng v 25 n 3 Mar 1986 p 409-414

Davison, D.A. and Meyer, G.E., (1984), Plant architecture measurements using microprocessor-based video imagery, ASAE Paper No. 84-5531, St. Joseph, MI:ASAE.

Dengel, Andreas (1989), "Automatic visual classification of printed documents", Proc Int Workshop Ind Appl Machine Intell Vision Proceedings of the International Workshop on Industrial Applications of Machine Intelligence and Vision Apr 10-12 1989 Tokyo, Jpn, p 276-281

Dixon, Mark J. Gregory, Paul J.(1986),"hybrid expert systems in image analysis",

Proceedings of SPIE - The International Society for Optical Engineering Applications of Artificial Intelligence IV. Presented at the Third International Symposium on Optical and Optoelectronic Applied Sciences and Engineering. v 657 1986 Innsbruck, Austria, p 9-12

Doyle, W., (1962), Operations useful for similarity-invariant pattern recognition, J. ACM 9, pp259-267

Duane, G. S.(1986),"knowledge-based segmentation of road scenes", Proceedings of SPIE - The International Society for Optical Engineering Applications of

Digital Image Processing IX. v 697 1986 San Diego, CA, USA,

p 250-259

Dubois, D. and H. Prade, (1980), Fuzzy Sets and Systems, Academic Press, INC, New York

Duda, R. O. and Hart, P.E., (1972), Use of the Hough Transformation to Detect lines and Curves in Pictures, Communications of the Ass. Comput. Mach. Vol. 15, pp11-15.

Duda, R. O. and Hart, P.E., (1973), Pattern Classification and Scene Analysis, Wiley, New York .

Duda, R. O. and P. E. Hart, (1973), Pattern Classification and Scene Analysis, John Wiley & Sons,Inc.

REFERENCES AND BIBLIOGRAPHY

Dunham, J. G., (1986), Optimum uniform piecewise linear approximation of planar curves, IEEE trans. Pattern Analysis and Machine Intelligence, vol. 8, pp57-75.

Dutta Majumder, D. (1988), "Computer vision and knowledge based computer systems", J Inst Electron Telecommun Eng v 34 n 3 May-Jun 1988, p230-245

Freeman, H,and Davis, L.S., (1977), A corner-finding algorithm for the chain-coded curves, IEEE Trans. Comput., Vol. C-26, pp297-303.

Foley, D.H. and J.W. Sammon, (1975), An optimal set of discriminant vectors, IEEE Trans. Compt.,vol. C-24, pp281-289.

Fu, K.S., (1976), Digital Pattern Recognition, Springer-verlag, ISBN 3-540-07511-9.

Gunasekaran, S., Cooper, T M., Berlage, A.G. and Krishnan, (198'), Image processing for stress cracks in corn kernels, Transactions of the ASAE, vol. 30(1), pp 266-271.

Goodrum, J.W. and Elster, R.T., (1992), Machine vision for crack detection in rotating eggs, American Society of Agricultural Engineers, vol. 35(4), pp1323-1328

Ghat, S.R., Evans, M.D., Kvien, C.K., and Rucker, K.S., (1993), Maturity detection in peanuts (aprachis hypogaea l.) using machine vision, Transactions of the ASAE, vol. 36(6), pp1941-1947

REFERENCES AND BIBLIOGRAPHY

Gregory, P. J. Taylor, C. J.(1984), "knowledge based models for computer vision",

Proceedings of the 4th International Conference on Robot Vision and Sensory Controls. 1984 London, Engl, p325-330

Hogg, D.C. (1993), Shape in machine visio. Image and Vision Computing,vol 11 no6, pp309-316.

Horn, B. K. P., (1986), Robot Vision, MIT Press, ISBN 0-262-08159-8.

Hough, P.V.C., (1962), Method and means for recognizing complex patterns, U.U. Patent 3,069,654.

Howarth, M. S. and J. R. Brandon, (1992) Estimation of tip shape for carrot classification by machine vision, J. Agric. Engng Res., vol. 53,pp123-139.

James, M. (1987), Pattern Recognition, BSP Professional Books.

Jamshidi, M., (1990), Sensors and interfacing in robotics and manufacturing, Robotic & Computer Integrated Manufacturing, vol.7,No 314.

Kandeh and Abraham, (1982), Fuzzy Techniques in Pattern Recognition, John Wiley & Sons.

Kandel, A., (1982), Fuzzy Techniques in Pattern Recognition, John Wiley & sons ISBN 0-471-09136-7.

Kelly, M.D. (1971), Edge detection in pictures by computer using planning, Machine Intelligence, B.Meltzer and D.Michie eds.,vol. 6 pp379-409,Univ. Press.

Kotoh, K.and Hiramatsu, K. (1973), A representation of pattern class using the fuzzy sets, Syst. Comput. Controls, pp1-8.

Kovasznay, L.S.G. and Joseph, H.M.(1955), Image processing, Proc. IRE43, pp560- 570

Kranzler, G.A., DeVoe, D.R. and Gentry, J. P. (1984), Opto-electronics and image processing for seedling counting in tree nursery beds, ASAE Paper No. 84-1091, St. Joseph, MI:ASAE

Krzyzak, A., T. Kasvand, C. Y. Suen, Eds. (1989), Computer Vision and Shape Recognition, World Scientific Publishing Co. Pte. Ltd. ISBN 9971-50-862-1

Krzyzak,A., W.Dai and C.Y. Suen, (1991), On the Recognition of Handwritten Characters Using Neural Networks, Pattern Recognition: Architecture, Algorithm & Application, World Scientific Co., Eds by R.Plamondon & H.D. Cheng ,pp115-135

Lieberman, L., (1979) Automatic computer analysis of images for recognition, inspection, verification, in Advances in Digital Image Processing, The IBM Research Symposia Series, Plenum Press,NY.

Ling, P.P. and Searcy, S.W., (1991), Feature extraction for a machine vision-based shrimp deheader, Transactions of the ASAE, vol. 34(6), pp2631-2636

Lee, H.C., (1981), A Computer system for Generating Object Description, PhD Thesis, Purdue University.

REFERENCES AND BIBLIOGRAPHY

Lee, E.T., Application of Fuzzy Languages to Pattern Recognition, Kybernetes,6, pp167-173.

Lee, E. T. (1975), Shape - oriented chromosome classification, IEEE Trans. Syst.,Man,Cybern, Vol. 5, pp629-623.

Lemmer, John F. Mitchell, Brian T. (1984), " interactive aids for the development of computer vision rule bases", Proceedings - PECORA 9. Spatial Information Technologies for Remote Sensing Today and Tomorrow, p249-257, 1984 Sioux Falls, ND, USA

Landau, U. M., (1987), Estimation of a circular arc center and its radius, Computer Vision,Graphics and Image Processing, Vol 38, pp317-326.

MacVicar-Whelcun, P.J., Fuzzy sets, the concept of height and the hedge very, IEEE Trans. Syst., Man. Cybern. ,8, p507-511

McCarthy, J., 1958,"Mechanization of Thought Processes," Proc. Symposium of National Physical Laboratory. London, Nov. p77-84.

Miller,B. K.,Delwiche,M. J.(1991), "Peach Defect Detection with Machine Vision", Transactions of the ASAE, vol. 34(6), pp 2588-2599.

Mucciardi, A.N. and E.E. Gose, (1971), A comparison of seven techniques for choosing subsets of pattern recognition properties, IEEE Trans. Compt.,vol. C-20,pp1023-1031.

REFERENCES AND BIBLIOGRAPHY

Nakagawa, Y. and Rosenfeld, A., (1979), A note on polygonal and elliptical approximation of mechanical parts, Pattern Recognition, vol. 11, pp133-142, Pergamon Press Ltd.

Narasimhan, R. and J.P. Fornango, (1963) Some further experiments in the parallel processing of picture, IEEE Tran. Electron. Comput. EC-12, pp748-750

Nazif, Ahmed M. Levine, Martin D. (1984),"low level image segmentation: an expert system", IEEE Trans Pattern Anal Mach Intell v PAMI-6 n 5 Sep 1984 p 555-577.

Negoita, C. V., (1985), Expert systems and Fuzzy Systems, The Benjamin Cummings Publishing Company, Inc

Neumann, Bernd (1985), "vision systems: state-of-the-art and prospect", Artificial Intelligence: Towards Practical Application, Proceedings of the Joint Technology Assessment Conference. 1985, p 51-61

Okamura, N.K., Delwiche, M.J., and Thompson, J.F., (1993), Raison grading by machine vision, Transactions of the ASAE, vol. 36(2), pp 485-492

Pavlidis, T. and Horowitz, (1974), Segmentation of plane curves, IEEE Trans. Computers, Vol. C-23, No. 8.

Perez, J. C. Castanet, R. (1985),"intelligent robot simulation system: the vision guided robot concept", Proceedings - COMPINT 85: Computer Aided Technologies. 1985 Montreal, Que, Can, p489-492

Pham, D.T. and Hafeez, K., (1992) Fuzzy model of a robot sensor for locating three dimensional objects, Robotica Vol 10, pp555-562.

Prewitt, J.M.S. and M.L. Mendelsohn, (1966), The analysis of cell images, Ann. N.Y. Acad. Sci. 128, pp 1033-1053

Radig, B.; Eckstein, W.; Klotz, K.; Messer, T.; Pauli, J. (1992), Automatization in the Design of Image Understanding Systems, Lecture notes in computer science,Vol604, p35-45.

Ramer,U., (1972), An iterative procedure for the polygonal approximation of plane curves, Comput. Graph. Image Proc., pp244-256.

Rigney, M.P., Brusewiz, and Kranzler, G. A., (1992), Asparagus defect inspection with machine, Transactions of the ASAE, vol.35(6), pp1873-1878.

Rehkugler, G.E. and Throop, J.A., (1987), Image processing algorithm for apple defect detection, ASAE Paper No. 87-3041, St. Joseph, MI:ASAE.

Roberts, L. G. (1965), Machine perception of three dimensional solids, in Optical and Electrooptical Information Processing, J. T. Tippett et al. eds, pp159-197, MIT Press.

Rossol, L., (1983), computer vision in industry, ROBOT VISION, Pugh, A. ed., IFS Ltd, UK

Rosenfeld,A.,and Weszka,J. S., (1975), An improved method of angle detection on the digital curves, IEEE trans. Comput.,vol. c-24, pp940-941.

Rosenfeld,A.,and Kak,A. C., (1976), Digital Picture Processing, Academic Press.

REFERENCES AND BIBLIOGRAPHY

Rosenfeld,A., and Johnston, E., (1973), Angle detection on digital curves, IEEE trans. Comput. vol. C-22, pp875-878.

Sarkar, N and Wolfe, R.R., (1985), Feature extraction techniques for sorting tomatoes by computer vision, Transactions of the ASAE, vol.28(3), pp970- 979.

Sankar, P. V. and Sharma, C. V., (1978) A parallel procedure for the detection of dominant points on a digital curve, Comput. Graphic Image Processing, vol. 7, pp403-412,1978.

Sawyer, B., Foster, D., (1986), Programming Expert systems in Pascal, John Wiley & Sons,Inc

Shen, H.C. and R. Pilkey, (1991), An adaptive approach to feature selection as applied to a texture classification problem, Architecture, Algorithm & Application, World Scientific Co., Eds by R. Plamondon &H.D. Cheng pp249-265.

Simonton, W. and Pease, J., (1993), Orientation independent machine vision classification of plants, Journal of Agricultural Engineering Research 1989, vol. 54, pp231-243.

Solinsky, James Clark (1986), "use of expert systems in machine vision recognition", Vision '86 - Conference Proceedings. 1986 Detroit, MI, Engl, p4.139-4.158

Staugaad, A. C., (1987), Robotics and AI, Prentice-hall,Inc

REFERENCES AND BIBLIOGRAPHY

Stucki, P., (1979), Advances in Digital Processing, The IBM Research Symposia Series, Plenum Press, ISBN 0-306-40314-5

Sutton, J.F. and McLendon, B. D. (1985), Development of an optoelectronic pine tree seedling counter, ASAE Paper No. 85-3044, St. Joseph, MI:ASAE

Tamas, A., (1989), On circles and circular arcs recognition, in Computer Vision and Shape Recognition,World Scientific Publishing Co. Ltd.

Tech,C.H. and Chin, R. T.,(1989), On the detection of dominant points on digital curves, IEEE trans. Pattern Analysis and Machine Intelligence, vol. 11, No. 8.

Tsuji, S. and Motsumoto, F., (1977), Detection of elliplic and linear edges by searching two parameter spaces, Proc. 5th IJCAI, pp700-705.

Turing, A.M., 1951,"Computing Machinery and Intelligence", in Computers and Thought, Feigenbaum, E. and J. Feldman(eds),McGraw Hill, New York,1963, pp11-35,

Upchurch, B. L. and Throop, J. A., (1994), Effects of storage duration on defecting watercore in apples using machine vision, Transactions of the ASAE, vol. 37(2), pp483-486.

Waterman, D.A., Eds. (1978), Pattern-directed Inference Systems, Academic Press.

Wesley, Leonard P. (1986), "evidentiai knowledge-based computer vision", Opt Eng v 25 n 3 Mar 1986 p 363-379

Westerlind, E., (1988), Seed scanner, a computer-based device for determinations of other seeds by number in cereal seed, Seed Science and Tech., 16:289-297.

Wu, Q. M. and Rodd, M. G., (1992), Boundary feature extraction and structural verification of objects, Advances in Machine Vision, pp35-62, World Scientific Pub[illegible] Co. Ltd.

Xiao, Q. and H. Raafat, (1991), Combining Statistical and Structural Information for Fingerprint Image Processing, Classification and Identification, Pattern Recognition: Architecture,Algorithm & Application, World Scientific Co., Eds by R.Plamond[illegible] H.D. Cheng ,pp 335-354

You, J. and H.A. Cohen, (1[illegible]1), Image segmentation by texture using tuned convolutions, in Pattern Recognition: Architecture, Algorithm & Application, World Scientific Co., Eds by R. Plamondon &H.D. Cheng, pp187-208

Zadeh, L. A., (1976), fuzzy sets and their application to pattern classification and cluster analysis, Memo UCB/ERL M-60, Univ. California, Berkeley.

Zusnes,L., (1970), Visual perception of form, Academic Press,New York.

APPENDIX A

C Source Code for the Knowledge Based System

File Name: imagep.c

```
/* Functions of performing image processing operations */

#include <v.h>
#include <math.h>
#include <termio.h>
#include        <conf.h>
#include <stdio.h>
#include <macro.h>
#include <imagep.h>

extern fnt_t                    fnt14x8, fnt8x6;
#define  FNT0                   (&fnt8x6)
#define  FNT1                   (&fnt14x8)

#define         Delt_angle      0.15    /* 8.6 degree */
#define         R_angle         0.1

extern unsigned long    rtclk();
extern int gen_display(int flg, but_t *b, win_t *fbu, win_t *w );
extern int obj_display(int flg, but_t *b, win_t *fbu, win_t *w );
extern int obj_analysis(int flg, but_t *b, win_t *fbu, win_t *w );
extern int add_blob(int flg,but_t *b, win_t *fbu, win_t *w);
extern int Dt_lnum, Feed_back;

win_t   fbu[4];

FILE    *io;
char    *filename;
char    *io_key;
int     num_merge,b_inf;

void getstr(str,n)
char *str;
int n;
{
        --n;
        read(0,str,n);
        for (; *str != '\n' && n--; str++ );
                *str ='\0';
}

setupluts()
{
```

```
        Vmkmap(LUT_F2D,12);
}

double distance( int u1, int v1,int u2,int v2)
{
        double dst;
        dst = sqrt( ((double)(v2-v1))*((double)(v2-v1))+((double)(u2-u1))*((double)(u2-u1)));
        return (dst);
}

#define         SMSEG 4

double  angle_compute( int u1, int v1, int u2, int v2)
{
        double  bta;

        if ( y2 -y1 > 0 )
        {
                if ( abs(x2-x1)< SMSEG ) {
                        bta = PI/2.0;
                        return (bta);
                }
                if ( x2 - x1 > 0 ) {
                        bta = atan( (double)(y2-y1)/(double)(x2-x1));
                        return (bta);
                }
                if ( x2 - x1 < 0 ) {
                        bta = PI + atan( (double)(y2-y1)/(double)(x2-x1));
                        return (bta);
                }
        }

        if ( y2 -y1 < 0 )
        {
                if ( abs(x2-x1)< SMSEG ) {
                        bta = PI/2.0+PI;
                        return (bta);
                }
                if ( x2 - x1 > 0 ) {
                        bta = PI*2.0 + atan( (double)(y2-y1)/(double)(x2-x1));
                        return (bta);
                }
                if ( x2 - x1 < 0 ) {
                        bta = PI + atan( (double)(y2-y1)/(double)(x2-x1));
                        return (bta);
                }
        }
}

showbut(b,bcol,tcol)
        register but_t      *b;
{
        fnt_t                   *f = FNT1;
        register        r, c;
```

```
        Vfill(&b->b_win,bcol);
        r = b->b_win.w_row + (b->b_win.w_nrow - f->fn_nrow) / 2 + f->fn_ascent;
        c = b->b_win.w_col + b->b_win.w_ncol / 2;
        Gtextc(r,c,tcol,f,b->b_text);
        Vdispl(&b->b_win,0,b->b_win.w_row,b->b_win.w_col);
}

b_quit(flg,b)
        but_t   *b;
{
        switch (flg) {
        case F_RFSH: showbut(b,BUBCOL,BUTCOL); break;
        case F_EXEC: term  = 1; break;
        }
        return 0;
}


b_live(flg,b,fbu,w)
        but_t   *b;
        win_t  *fbu,*w;
{
        register  en, key;
        coor_t          svpos;

        switch (flg) {
        case F_RFSH: showbut(b,BUBCOL,BUTCOL); break;
        case F_EXEC:
                Gcursor(0);
                Gcursor(3,&svpos);
                showbut(b,BUHCOL,BUTCOL);
                for (en = 0; ; ) {
                        if (Gcursor(2))
                                if (!(key = Gcursor(3,0))) en = 1;
                                else if (en && key) break;
                        Vfill(&fbu[3],0);
                        Vxsnap(2,&fbu[3]);
                        Vreduc(&fbu[3],LUT_F2D,&w[0],2);
                        Vdispl(&w[0],0,w[0].w_row,w[0].w_col);
                }
                showbut(b,BUBCOL,BUTCOL);
                Gcursor(4,&svpos);
                Gcursor(1);
                break;
        }
        return;
}

#define         Dsd     10

find_corner_point()
{
        int     z,z1,z2,w,w1,w2,i,j,xtp,ytp,blank,pix_val,drow,dcol;

        i = 0;
```

```
Ncp = 0;
for ( j=1; j<5; j++)
{
        switch(j)
        {
                case 1:
                        z1 = bcol_st;
                        w1 = brow_st;
                        w2 = brow_end;
                        break;
                case 2:
                        z1 = brow_st;
                        w1 = bcol_st-Dsd;
                        w2 = bcol_end+Dsd;
                        break;
                case 3:
                        z1 = bcol_end;
                        w1 = brow_st;
                        w2 = brow_end;
                        break;
                case 4:
                        z1 = brow_end;
                        w1 = bcol_st-Dsd;
                        w2 = bcol_end+Dsd;
                        break;
        }
        if ( j==1 || j==2 )
        goto minus;
plus:
        printf("\n j=%d z1=%d,z2=%d w1=%d w2=%d",j,z1,z1+Dsd,w1,w2);
                blank = 0;
                pix_val = 0;
                xtp = 0;
                ytp = 0;
                for ( z = z1; z<z1+Dsd; z++ )
                {
                        for ( w=w1; w< w2; w++ )
                        {
                                if ( j==1 || j==3)
                                        pix_val = Vwix(&fbu[2],w,z);
                                else if ( j==2 || j==4)
                                        pix_val = Vwix(&fbu[2],z,w);
                                if ( pix_val == 255 ) {
                                if ( j==1 || j==3) {
                                        xtp = w; /* save as temptery */
                                        ytp = z;
                                }
                                if ( j==2 || j==4) {
                                        xtp = z;
                                        ytp = w;
                                }
                                if ( i==1 || j==2 )
                                goto find_cp;
                                else
                                goto next_z1;
```

```
                        }
                    }
                    blank++;        /* blank line */
                    if ( blank >5 || z==z1+Dsd-1 )
                    {
                        if ( xtp == 0 ) {
                            if ( j==1 || j==2)
                            goto next_sd;
                            if ( j==3 || j==4 )
                            goto minus;
                        }
find_cp:
                        if ( i>1 )
                        {
                            drow = xtp-cp[i].c_row;
                            dcol = ytp-cp[i].c_col;
                            if ( abs(drow) <4 && abs(dcol) <4 )
                            goto next_sd;
                            if ( i==3 ) {
                            drow = xtp-cp[1].c_row;
                            dcol = ytp-cp[1].c_col,
                            if ( abs(drow) <4 && abs(dcol) <4 )
                            goto next_sd;
                            }

                        }
                        i++;
                        cp[i].c_row = xtp;
                        cp[i].c_col = ytp;
                        goto next_sd;
                    }
next_z1: continue;
                }
minus:
        printf("\n j=%d z1=%d,z2=%d w1=%d w2=%d",j,z1,z1-Dsd,w1,w2);
                blank = 0;
                pix_val = 0;
                xtp = 0;
                ytp = 0;
                for ( z = z1; z>z1-Dsd; z-- )
                {
                    for ( w=w1; w< w2; w++ )
                    {
                        if ( j==1 || j==3)
                        pix_val = Vwix(&fbu[2],w,z);
                        if ( j==2 || j==4)
                        pix_val = Vwix(&fbu[2],z,w);
                        if ( pix_val == 255 ) {
                            if ( j==1 || j==3) {
                                xtp = w; /* save as temptery */
                                ytp = z;
                            }
                            if ( j==2 || j==4) {
                                xtp = z;
                                ytp = w;
```

```
                                        }
                                printf("\n(%d,%d)=%d",xtp,ytp,pix_val);
                                if ( j==3 || j==4 )
                                goto find_cp;
                                else
                                goto next_z2;
                                }
                        }
                        blank++;        /* blank line */
                        if ( blank >5 || z==z1-Dsd+1 )
                        {
                                if ( xtp == 0 ) {
                                        if ( j==1 || j==2)
                                        goto plus;
                                        if ( j==3 || j==4 )
                                        goto next_sd;
                                }
                                if ( i>1 )
                                {
                                        drow = xtp-cp[i].c_row;
                                        dcol = ytp-cp[i].c_col;
                                        if ( abs(drow) <4 || abs(dcol) <4 )
                                        goto next_sd;
                                        if ( i==3 ) {
                                        drow = xtp-cp[1].c_row;
                                        dcol = ytp-cp[1].c_col;
                                        if ( abs(drow) <4 && abs(dcol) <4 )
                                        goto next_sd;
                                        }
                                }
                                i++;
                                cp[i].c_row = xtp;
                                cp[i].c_col = ytp;
                                goto next_sd;
                        }
next_z2:        continue;
                }
next_sd:
                continue;
        }
        Ncp = i;
}

find_side_most( bnum)
{
        int     row_st,nrow,col_st,ncol,row,col,pix_val,dtb;
        row_st = bbox[bnum].w_row;
        nrow = bbox[bnum].w_nrow;
        col_st = bbox[bnum].w_col;
        ncol = bbox[bnum].w_ncol;
        dtb = DTB;
left_top:
        for ( row=row_st;row<row_st+nrow;row++)
        {
                for (col=col_st - dtb; col<col_st+dtb; col++)
```

```
                {
                        pix_val = Vwix(&fbu[2],row,col);
                        if ( pix_val == 255 )
                        {
                                left_t.c_row = row;
                                left_t.c_col = col;
                                goto left_bottom;
                        }
                }
        }
        dtb++;
        goto left_top;

left_bottom:
        dtb = DTB;
left_bt:
        for ( row=row_st + nrow; row > row_st; row--)
        {
                for (col=col_st - dtb; col<col_st+dtb; col++)
                {
                        pix_val = Vwix(&fbu[2],row,col);
                        if ( pix_val == 255 )
                        {
                                left_b.c_row = row;
                                left_b.c_col = col;
                                goto right_top;
                        }
                }
        }
        dtb++;
        goto left_bt;
right_top:
        dtb = DTB;
right_tp:
        for ( row=row_st;row<row_st+nrow;row++)
        {
                for (col=col_st+ncol + dtb; col>col_st+ncol-dtb; col--)
                {
                        pix_val = Vwix(&fbu[2],row,col);
                        if ( pix_val == 255 )
                        {
                                right_t.c_row = row;
                                right_t.c_col = col;
                                goto right_bottom;
                        }
                }
        }
        dtb++;
        goto right_tp;
right_bottom:
        dtb = DTB;
right_bt:
        for ( row=row_st + nrow; row > row_st; row--)
        {
                for (col=col_st+ncol + dtb; col>col_st+ncol-dtb; col--)
```

```
                {
                        pix_val = Vwix(&fbu[2],row,col);
                        if ( pix_val == 255 )
                        {
                                right_b.c_row = row;
                                right_b.c_col = col;
                                goto top_left;
                        }
                }
        }
        dtb++;
        goto right_bt;

top_left:
        dtb = DTB;
top_lt:
        for ( col= col_st; col< col_st + ncol; col++)
        {
                for ( row = row_st - dtb; row < row_st +dtb; row++ )
                {
                        pix_val = Vwix(&fbu[2],row, col);
                        if ( pix_val == 255 )
                        {       top_l.c_row = row;
                                top_l.c_col = col;
                                goto top_right;
                        }
                }
        }
        dtb++;
        goto top_lt;

top_right:
        dtb = DTB;
top_rt:
        for ( col= col_st + ncol; col> col_st; col-- )
        {
                for ( row = row_st - DTB; row < row_st +dtb; row++ )
                {
                        pix_val = Vwix(&fbu[2],row, col);
                        if ( pix_val == 255 )
                        {       top_r.c_row = row;
                                top_r.c_col = col;
                                goto bottom_left;
                        }
                }
        }
        dtb++;
        goto top_rt;
bottom_left:
        dtb = DTB;
bottom_lt:
        for ( col= col_st; col< col_st + ncol; col++)
        {
                for ( row = row_st +nrow + dtb; row>row_st + nrow -dtb; row--)
                {
```

```
                        pix_val = Vwix(&fbu[2],row, col);
                        if ( pix_val == 255 )
                        {       bottom_l.c_row = row;
                                bottom_l.c_col = col;
                                goto bottom_right;
                        }
                }
        }
        dtb++;
        goto bottom_lt;
bottom_right:
        dtb = DTB;
bottom_rt:
        for ( col= col_st + ncol; col> col_st; col-- )
        {
                for ( row = row_st +nrow + dtb; row>row_st + nrow -dtb; row--)
                {
                        pix_val = Vwix(&fbu[2],row, col);
                        if ( pix_val == 255 )
                        {       bottom_r.c_row = row;
                                bottom_r.c_col = col;
                                return 0;
                        }
                }
        }
        dtb++;
        goto bottom_rt;

}

side_point_detection( int bnum)
{
        int     flg_st,flg_end,i,k,pix_val,j,ini,z1,z2;
        int     row,col,row_st,row_end,col_st,col_end,col_t,row_t;
        int     wd;
        coor_t  tp1,tp2;
        row_st = bbox[bnum].w_row;
        row_end = bbox[bnum].w_nrow + row_st;
        col_st = bbox[bnum].w_col;
        col_end = bbox[bnum].w_ncol + col_st;
        flg_st = 0;
        flg_end = 0;
        Ak=0;
        Bk=0;
        Ck=0;
        Dk=0;
        for( j=1;j<5; j++)
        {
                k=0;
                flg_st=0;
                flg_end=0;
                switch(j)
                {
                        case 1:
                                ini=col_st;
```

```
                                z1=row_st;
                                z2=row_end;
                                break;
                        case 2:
                                z1=row_st;
                                z2=row_end;
                                ini=col_end;
                                break;
                        case 3:
                                z1=col_st+3;  /* avoid to fall in A or C */
                                z2=col_end-3;
                                ini=row_st;
                                break;
                        case 4:
                                z1=col_st+3;
                                z2=col_end-3;
                                ini=row_end;
                                break;
                }
        for ( row=z1; row< z2+3; row++ )
        {
                for ( i=0; i<4;i++)          /* within [+3, -3] */
                {
                        col=ini+i;
find_sdp:
                        if ( j<3 )
                        pix_val=Vwix(&fbu[2],row,col);
                        else
                        pix_val=Vwix(&fbu[2],col,row);
                        if ( pix_val== 255)
                        {
                                row_t=row;
                                col_t=col;
                                flg_end=0;
                                if(flg_st==0)
                                {
                                        flg_st=1;
                                        k++;
                                        switch(j)
                                        {
                                        case 1:
                                                A[k].c_row=row;
                                                A[k].c_col=col;
                                                break;
                                        case 2:
                                                C[k].c_row=row;
                                                C[k].c_col=col;
                                                break;
                                        case 3:
                                                B[k].c_row=col;
                                                B[k].c_col=row;
                                                break;
                                        case 4:
                                                D[k].c_row=col;
                                                D[k].c_col=row;
```

```
                                        break;
                                }
                                break,
                        }
                        if( flg_st==1 )
                        break;
                }else {
                        if( col<=ini )
                        continue;
                        else
                        {
                        col=ini-i;
                        goto find_sdp;
                        }
                }
        }

        if(flg_st==1 && pix_val==0) /* when blank points >2 endpoint come out*/
        {
                flg_end++;
                if ( flg_end >2 )
                {
                        k++;
                switch(j)
                {
                case 1:
                        A[k].c_row=row_t;
                        A[k].c_col=col_t;
                        break;
                case 2:
                        C[k].c_row=row_t;
                        C[k].c_col=col_t;
                        break;
                case 3:
                        B[k].c_row=col_t;
                        B[k].c_col=row_t;
                        break;
                case 4:
                        D[k].c_row=col_t;
                        D[k].c_col=row_t;
                        break;
                }
                flg_st=0;
                flg_end=0;
                continue;
                } else
                continue;
        } else
        continue;
}
switch(j)
{

        case 1:
                Ak=k;
```

```
                    break;
                case 2:
                    Ck=k;
                    break;
                case 3:
                    Bk=k;
                    break;
                case 4:
                    Dk=k;
                    break;
        }
        }
/* produce the corner points cp[i] and the number Ncp*/
        Ncp=0;
        i=0;
for( j=1; j<5; j++ )
{
        switch(j)
        {
                case 1:
                    k = Ak;
                    break;
                case 2:
                    k = Bk;
                    break;
                case 3:
                    k = Ck;
                    break;
                case 4:
                    k = Dk;
                    break;
        }
        switch(k)
        {
                case 0: break;
                case 1:
                        switch ( j )
                        {
                                case 1:
                                        tp1.c_row = A[1].c_row;
                                        tp1.c_col = A[1].c_col;
                                        break;
                                case 2:
                                        tp1.c_row = B[1].c_row;
                                        tp1.c_col = B[1].c_col;
                                        break;
                                case 3:
                                        tp1.c_row = C[1].c_row;
                                        tp1.c_col = C[1].c_col;
                                        break;
                                case 4:
                                        tp1.c_row = D[1].c_row;
                                        tp1.c_col = D[1].c_col;
                                        break;
                        }
```

```
                        i++;
                        cp[i].c_row = tp1.c_row;
                        cp[i].c_col = tp1.c_col;
                        break;
                case 2:
                        switch ( j )
                        {
                                case 1:
                                        tp1.c_row = A[2].c_row;
                                        tp1.c_col = A[2].c_col;
                                        tp2.c_row = A[1].c_row;
                                        tp2.c_col = A[1].c_col;
                                        wd = tp2.c_row - tp1.c_row;
                                        break;
                                case 2:
                                        tp1.c_row = B[2].c_row;
                                        tp1.c_col = B[1].c_col;
                                        tp2.c_row = B[2].c_row;
                                        tp2.c_col = B[2].c_col;
                                        wd = tp2.c_col - tp1.c_col;
                                        break;
                                case 3:
                                        tp1.c_row = C[1].c_row;
                                        tp1.c_col = C[1].c_col;
                                        tp2.c_row = C[2].c_row;
                                        tp2.c_col = C[2].c_col;
                                        wd = tp2.c_row - tp1.c_row;
                                        break;
                                case 4:
                                        tp1.c_row = D[2].c_row;
                                        tp1.c_col = D[2].c_col;
                                        tp2.c_row = D[1].c_row;
                                        tp2.c_col = D[1].c_col;
                                        wd = tp2.c_col - tp1.c_col;
                                        break;
                        }
/* modify cp[i] change later */
                        if ( abs(wd) <5 )
                        {
                        i++;
                        cp[i].c_row = ( tp1.c_row + tp2.c_row )/2;
                        cp[i].c_col = ( tp1.c_col + tp2.c_col )/2;
                        } else {
                        i++;
                        cp[i].c_row = tp1.c_row;
                        cp[i].c_col = tp1.c_col;
                        i++;
                        cp[i].c_row = tp2.c_row;
                        cp[i].c_col = tp2.c_col;
                        }
                        break;
        }
}
        Ncp = i;
        return;
```

```
}

chord_height()
{
        int     row_st,row_end,row,col,pix_val,i,flg_rpt=0,rpt=0,row0;
        int     r1,c1,r2,c2;
        r1 = x1;
        c1 = y1;
        r2 = x2;
        c2 = y2;
        if ( y2<y1 )
        {
                x1 = x2;
                y1 = y2;
                x2 = r1;
                y2 = c1;
        }
        kxy = ((float)(x2 -x1))/((float)(y2-y1));
        afa = atan (kxy);

        Vdispl(&fbu[2],0,0,0);
        xlm = (x2+x1)/2;
        ylm = (y2+y1)/2;
        printf("\np1(%d,%d),p2(%d,%d) lm(%d,%d)
am(%d,%d)",x1,y1,x2,y2,xlm,ylm,xam,yam);
        G_line(0,xo,yo,xlm,ylm,RED);
        if ( flg_dir ==0 ) {
                row_st=brow_st-4;
                row_end=x1;
                if ( x2>x1)
                row_end=x2+4;
        }
        if ( flg_dir == 1) {
                row_end=brow_end+2;
                row_st=x2-4;
                if ( x2>x1)
                row_st = x1;
        }

        printf("\nbrow_st=%d brow_end=%d",brow_st,brow_end);
        printf("\nrow_st=%d row_end=%d flg_dir=%d",row_st,row_end,flg_dir);
        pix_val = 0;
/* center of outline : intersection between vertical line and outline*/
outline_c:
        for ( i=1; ylm +i <bcol_end; i++)
        {
                if ( ylm-i <bcol_st )
                break;
                col = ylm +i;
                rpt=0;
                row0 = xlm + (int)((double)(i)*tan(PI/2.0+afa)+0.5); /**V**/

                if ( row0> row_end || row0 < row_st ) {
                        if ( rpt==0 )
                        goto rpt_1;
```

```
                    else
                    continue;
                }
                row=row0;
on_arc:
                G_line(0,x2,y2,row,col,GREEN);
                pix_val = Vwix(&fbu[2],row,col);
/*              printf("pxv=%d",pix_val);*/
                if ( pix_val == 255 )
                {
                xam = row;
                yam = col;
                printf("\n am(%d,%d)",xam,yam);
                break;
                }
                switch ( row-row0 )
                {
                        case 0:
                                row=row0+1;
                                printf("\nrow0+1=%d",row);
                                goto on_arc;
                        case 1:
                                row=row0-1;
                                printf(" row0-1=%d",row);
                                goto on_arc;
                        case -1:
                                row=row0+2;
                                printf("\nrow0+2=%d",row);
                                goto on_arc;
                        case 2:
                                row=row0-2;
                                printf("\nrow-2=%d",row);
                                goto on_arc;
                        case -2:
                                break;
                }
col_dec:
                if ( rpt==1 )
                continue;
rpt_1:          col=ylm-i;
                row0 = xlm - (int)((double)(i)*tan(PI/2.0-afa) +0.5 ); /**V**/
                rpt=1;
                row=row0;
                goto on_arc;
        }
 /* if search fails, shift the vertical line ylm a bit right*/

        if( pix_val != 255 )
        {
                if ( flg_rpt==0) {
                        ylm=ylm+1;
                        flg_rpt=1;
                        goto outline_c;
                }
                if ( flg_rpt ==1 ) {
```

```
                        ylm=ylm-2;
                        flg_rpt=2;
                        goto outline_c;
                }
                if ( flg_rpt==2) {
                        printf("\n Can not find the middle point of the primitive.\n");

                }
        }
                pix_val = Vwix(&fbu[2],x1,y1);
                printf("(x1,y1)pxv=%d",pix_val);
                pix_val = Vwix(&fbu[2],x2,y2);
                printf("(x2,y2)pxv=%d",pix_val);
        dd=distance(x1,y1,x2,y2);
        hh = ((double)(abs(xam-xlm)))/cos(afa);
        belta = atan( hh/dd*2.0);
        G_line(0,xo,yo,xam,yam,RED);
printf("\n arc center(%d,%d) Line center(%d,%d) \n dd=%f hh = %f",xam,yam,xlm,ylm,dd,hh);
printf("\n Angle:afa=%f  belta=%f  tg(90+a)=%f",afa,belta,tan(PI/2.0+afa));
        if ( c2< c1 )
        {
                x1 = r1;
                y1 = c1;
                x2 = r2;
                y2 = c2;
        }
        return;
}


#define         Delt_len 0.16

circle_detection( int bnum )
{
        int     i,j,pix_val,row,col,num_out_cir=0,unsym;
        int     col_st,col_end,row0,n=0;
        double  theta,Delt_theta=PI/36.0,rr[73],r1,r2,o_cp[4];

        Rx = 0;
/* premiary check , symmetry test*/
        x1=xo;
        y1=yo;
        o_cp[1] = 0;
        for(j=1;j<Ak+1;j++) {
        x2=A[j].c_row;
        y2=A[j].c_col;
        o_cp[1] = o_cp[1]+ distance(x1,y1,x2,y2);
        }
        o_cp[1] = o_cp[1]/Ak;
        o_cp[2] = 0;
        for(j=1;j<Bk+1;j++) {
        x2=B[j].c_row;
        y2=B[j].c_col;
        o_cp[2] = o_cp[2]+ distance(x1,y1,x2,y2);
        }
```

```
        o_cp[2] = o_cp[2]/Bk;
        o_cp[3] = 0;
        for(j=1;j<Ck+1;j++) {
        x2=C[j].c_row;
        y2=C[j].c_col;
        o_cp[3] = o_cp[3]+ distance(x1,y1,x2,y2);
        }
        o_cp[3] = o_cp[3]/Ck;
        o_cp[4] = 0;
        for(j=1;j<Dk+1;j++) {
        x2=D[j].c_row;
        y2=D[j].c_col;
        o_cp[4] = o_cp[4]+ distance(x1,y1,x2,y2);
        }
        o_cp[4] = o_cp[4]/Dk;
        Rx = (o_cp[1] + o_cp[2] + o_cp[3] + o_cp[4] )/4.0;
        printf("\n Rx=%f", Rx);
        unsym = 0;
        for ( i=2; i<5; i++ )
        {
                printf("\n o_cp= %f",o_cp[i]);
                if ( fabs( o_cp[i] - o_cp[1] )/o_cp[1] > Delt_len )
                unsym++;
        }
        if ( unsym>1 )
                goto un_sym;
/* Circle Check */
        Vdispl(&fbu[2],0,0,0);
        for ( i= 0; i<73; i++)
                rr[i]=0;
/* compute the presumed circle radius */
/*      r1 = ( right_t.c_col - left_t.c_col )/2;
        r2 = ( bottom_l.c_row - top_l.c_row)/2;*/
/* check points along the step Delt_theta */
        for ( i=1; i<73; i++)
        {
                theta = Delt_theta*((double)i);
                if ( i<18 || i >54 ) {
                        col_st = bcol_st-4;
                        col_end = yo;
                }
                if ( i > 18 && i <54 ) {
                        col_st = yo;
                        col_end= bcol_end+4;
                }
                if ( i==18 ) {
                        col = yo;
                        for ( j=0; j<4;j++) {
                        row = brow_st+j;
                        pix_val=Vwix(&fbu[2],row,col);
                        if ( pix_val==255) {
                                rr[i] = xo-row;
                                break;
                        }
                        row = brow_st -j;
```

```
                pix_val=Vwix(&fbu[2],row,col);
                if ( pix_val==255) {
                        rr[i] =xo-row;
                        break;
                }
                }
                goto on_curve;
        }
        if ( i==54 ) {
                col = yo;
                for ( j=0; j<4;j++) {
                row = brow_end+j;
                pix_val=Vwix(&fbu[2],row,col);
                if ( pix_val==255) {
                        rr[i] = row-xo;
                        break;
                }
                row = brow_end -j;
                pix_val=Vwix(&fbu[2],row,col);
                if ( pix_val==255) {
                        rr[i] =row -xo;
                        break;
                }
                }
                goto on_curve;
        }
        if ( i==36 ) {
                row = xo;
                for ( j=0; j<4;j++) {
                col= bcol_end+j;
                pix_val=Vwix(&fbu[2],row,col);
                if ( pix_val==255) {
                        rr[i] = col-yo;
                        break;
                }
                col = bcol_end -j;
                pix_val=Vwix(&fbu[2],row,col);
                if ( pix_val==255) {
                        rr[i] = col-yo;
                        break;
                }
                }
                goto on_curve;
        }
        if ( i==72) {
                row = xo;
                for ( j=0; j<4;j++) {
                col= bcol_st+j;
                pix_val=Vwix(&fbu[2],row,col);
                if ( pix_val==255) {
                        rr[i] =yo- col;
                        break;
                }
                col = bcol_st -j;
                pix_val=Vwix(&fbu[2],row,col);
```

```
                        if ( pix_val==255) {
                                rr[i] =yo- col;
                                break;
                        }
                        }
                        goto on_curve;
                }
                for ( col = col_st; col < col_end+2; col++)
                {
                        row0 = xo + (int)(((double)(col-yo))*tan(theta));
                        if ( row0 > brow_end+4 || row0 < brow_st-4 )
                        continue;
                        row = row0;
chk_p:
                        pix_val = Vwix(&fbu[2], row,col);
                        if ( pix_val == 255)
                        {
                                rr[i] = fabs ( (double)(row-xo)/sin(theta));
on_curve:
                                G_line(0,xo,yo,row,col,GREEN );
                                break;
                        }
                        if ( pix_val ==0 && row ==row0 ) {
                                row ++;
                                goto chk_p;
                        }
                        if ( pix_val ==0 && row>row0) {
                                row = row0 -1;
                                goto chk_p;
                        }
                }
        }
        ro = rr[0];
        for ( i=1; i<73; i++) {
                if (rr[i] == 0 ) {
                        n++;
                        continue;
                }
                ro = ro + rr[i];
        }
        ro = ro/((double)(72-n));
        for ( i=1; i<73; i++)
        {
                if ( rr[i] == 0 )
                        continue;
                if ( (fabs( rr[i] - ro ))/ro > Delt_r ) {
                        printf("\nrr[%d]-ro=%f",i,rr[i]-ro);
                        num_out_cir++;
                }
        }
        printf("\n Presumed circle radius ro =%f ",ro);
        printf("\n %d points exceeds the threshold of the presumed circle",num_out_cir);
        if ( num_out_cir > 5)
        goto Not_circle;
Is_circle:
```

```
        printf("\n IT is a CIRCLE, center is (%d,%d) Radius=%f",xo,yo,ro);
        flg_circle = 1;
        return 1;
Not_circle:
        printf("\n %d points exceeds the threshold of the presumed circle",num_out_cir);
        printf("\n Presumed circle radius ro =%f ",ro);
un_sym:
        printf("\n It is NOT a Circle.");
        flg_circle = 0;

        return 0;
}

#define         Dtheta  0.0349066               /* 2 degree */

arc_detection()
{
        coor_t  pp[4];
        int     col_st,col_end,row,col,pix_val,i,j,k;
        int     xc0,yc0,col0,row0,num_out_arc=0;
        double  h_p,D_p,R_p,R0,r1,r2,rm,r,Rx;
        double  theta,theta1,theta2,theta_end;
        int     xt1,yt1,xt2,yt2;

        flg_arc_low = 0;
        xt1 = x1;
        yt1 = y1;
        xt2 = x2;
        yt2 = y2;
        if ( y2 <y1 )
        {
                x1 = x2;
                y1 = y2;
                x2 = xt1;
                y2 = yt1;
        }
/* middle point of pp */
        Rx = 0;
        chord_height();
        flg_arc=0;
        pp[0].c_row = ( xam + xlm )/2;
        pp[0].c_col = ( yam + ylm )/2;
/*      Vdispl(&fbu[2],0,0,0); */
/* search pp[1] pp[2] */
                col_st = pp[0].c_col;
                col_end = bcol_end;
ppp:            for ( col=col_st+1; col<col_end; col++)
                {
                        row0 = pp[0].c_row +(int)((double)(col-pp[0].c_col)*tan(afa));
                        row=row0;
srch_pp1:
                        if ( row > brow_end+2 || row < brow_st-4 )
                        continue;
                        pix_val = Vwix(&fbu[2],row,col);
/*                              G_line(0,xlm,ylm,row,col,GREEN);*/
```

```
                        if ( pix_val == 255 )
                        {
                                pp[2].c_row = row;
                                pp[2].c_col = col;
                                printf("\npp[2](%d,%d)",row,col);
/*                              G_line(0,pp[0].c_row,pp[0].c_col,row,col,RED);*/
                                break;
                        }
                        if ( row ==row0-1 )
                        continue;
                        if ( row==row0 ) {
                                row=row0+1;
                                goto srch_pp1;
                        }
                        if ( row==row0+1 ) {
                                row=row0-1;
                                goto srch_pp1;
                        }
                }

/*              if ( pix_val==0 ) {
                pp[0].c_row++;
                goto ppp;
                }*/

                col_end = pp[0].c_col;
/*              col_st = y1; */
                for ( col=col_end-1; col>bcol_st-2; col--)
                {
                        row0 = pp[0].c_row +(int)((double)(col-pp[0].c_col)*tan(afa));
                        row = row0;
srch_pp2:
                        if ( row > brow_end+2 || row < brow_st-4 )
                        continue;
                        pix_val = Vwix(&fbu[2],row,col);
/*                              G_line(0,xlm,ylm,row,col,GREEN);*/
                        if ( pix_val == 255 )
                        {
                                pp[1].c_row = row;
                                pp[1].c_col = col;
                        printf("\npp[1](%d,%d)",row,col);
                /*              G_line(0,pp[0].c_row,pp[0].c_col,row,col,RED); */
                                break;
                        }
                        if ( row ==row0-1 )
                        continue;
                        if ( row==row0 ) {
                                row=row0+1;
                                goto srch_pp2;
                        }
                        if ( row==row0+1 ) {
                                row=row0-1;
                                goto srch_pp2;
                        }
```

```
                }
        G_line(0,pp[1].c_row,pp[1].c_col,pp[2].c_row,pp[2].c_col,BLUE),
        G_line(0,x1,y1,x2,y2,GREEN);

        h_p = 0.5*hh;
        D_p = fabs((double)(pp[2].c_row-pp[1].c_row)/sin(afa));
        R_p = (h_p*h_p + D_p*D_p/4.0 )/h_p/2.0;
        R0 =  ( hh*hh + dd*dd/4.0 )/hh/2.0;
/*      if ( flg_dir == 0 ) {
        pp[3].c_row = pp[0].c_row+(int)((R_p-h_p)*sin(afa+ PI/2.0));
        pp[3].c_col = pp[0].c_col+(int)((R_p-h_p)*cos(afa+ PI/2.0));
        xc0  = xlm + (int)((R0-hh)*sin(afa+ PI/2.0));
        yc0  = ylm + (int)((R0-hh)*cos(afa+ PI/2.0));
        } else {
        xc0  = xlm + (int)((R0-hh)*sin(afa - PI/2.0));
        yc0  = ylm + (int)((R0-hh)*cos(afa - PI/2.0));
        pp[3].c_row = pp[0].c_row+(int)((R_p-h_p)*sin(afa - PI/2.0));
        pp[3].c_col = pp[0].c_col+(int)((R_p-h_p)*cos(afa - PI/2.0));
        }       */
        if ( xam < xlm ) {
        pp[3].c_row = pp[0].c_row+(int)((R_p-h_p)*fabs(sin(afa+ PI/2.0)));/* pp[3]circle
center*/
        xc0  = xlm + (int)((R0-hh)*fabs(sin(afa+ PI/2.0)));
        } else {
        xc0  = xlm - (int)((R0-hh)*fabs(sin(afa - PI/2.0)));
        pp[3].c_row = pp[0].c_row - (int)((R_p-h_p)*fabs(sin(afa - PI/2.0)));
        }

        if ( yam < ylm ) {
        pp[3].c_col = pp[0].c_col+(int)((R_p-h_p)*fabs(cos(afa+ PI/2.0)));
        yc0  = ylm + (int)((R0-hh)*fabs(cos(afa+ PI/2.0)));
        } else {
        yc0  = ylm - (int)((R0-hh)*fabs(cos(afa - PI/2.0)));
        pp[3].c_col = pp[0].c_col - (int)((R_p-h_p)*fabs(cos(afa - PI/2.0)));
        }
        xc = ( xc0 + pp[3].c_row )/2;
        yc = ( yc0 + pp[3].c_col )/2;
        G_line(0,x1,y1,xc,yc,RED);
        G_line(0,x2,y2,xc,yc,RED);
        G_line(0,x1,y1,xc0,yc0,RED);
        G_line(0,x1,y1,pp[3].c_row,pp[3].c_col,RED);
        R = ( R0 + R_p )/2.0;
        if ( xc < 0 || xc>512 || yc<0 || yc>512)
        goto arc_chk;
        G_line(0,pp[1].c_row,pp[1].c_col,pp[3].c_row,pp[3].c_col,BLUE);
        G_line(0,x1,y1,xc,yc,RED);
        G_line(0,x2,y2,xc0,yc0,GREEN);
/* check distance between centre and the points on the outline */
/* beween angle theta1 and theta2, step=Dtheta */
/*                |180
        atan>0  | atan<0
   360  -----|------>y 90
        atan<0  | atan>0
             +
                x 0
```

```
atan(x):[-PI/2,+PI/2]
*/
arc_chk:
        if ( abs(x1-xc)<4  ) {
                if ( y1<yc )
                theta1 = 1.5*PI;
                else
                theta1 = PI/2.0;
        } else {
                theta1 = atan((double)(y1-yc)/(double)(x1-xc));
                if ( x1<xc )
                        theta1 = theta1+PI;
                if ( x1>xc && y1<yc )
                        theta1 = theta1 + 2*PI;
        }
        if ( abs(x2-xc)<4  ) {
                if ( y2<yc )
                theta2 = 1.5*PI;
                else
                theta2 = PI/2.0;
        } else {
                theta2 = atan((double)(y2-yc)/(double)(x2-xc));
                if ( x2<xc )
                        theta2 = theta2+PI;
                if ( x2>xc && y2<yc )
                        theta2 = theta2 + 2*PI;
        }
        if ( theta1> theta2){
                theta=theta1;
                theta1=theta2;
                theta2=theta;
        }
        theta = atan((double)(y1-xam)/(double)(x1-yam));
        r1 = fabs((double)(x1-xc)/cos(theta1));
        r2 = fabs((double)(x2-xc)/cos(theta2));
        col_end =y2;
        col_st = y1;
        printf("\n theta1=%f theta2=%f",theta1,theta2);
/* arc in low position */
        if ( theta2 > PI && theta2<PI*2.0 )
        {
                if ( theta1<PI/2.0 )
                flg_arc_low = 1;
        }
        if ( theta2> PI*1.5 && theta2 <PI*2.0 )
        {
                if ( theta1 < PI )
                flg_arc_low = 1;
        }
        theta_end = theta2 - theta1;
        if ( flg_arc_low ==1 )
                theta_end = theta1-theta2+PI*2.0;
        for ( j=1: Dtheta*((double)(j))< theta_end; j++)
        {
```

```
                theta = theta1 + Dtheta*((double)(j));
                if ( flg_arc_low == 1 )
                        theta = theta2 + Dtheta*((double)(j));
                col0 = yc + (int)( R*sin(theta)); /* assumed arc */
/* theta near PI */
                if ( theta<PI+0.008726 && theta>PI-0.008726 )/*0.5degree,ie Pi/360 */
                {
                        row0 = xc-R;
                        col =col0;
                        goto srch_vt_dir;
                }
/* theta near PI*2 */
                if ( theta<PI*2.0+0.008726 && theta>PI*2.0-0.008726 )/*0.5degree,ie Pi/360 */
                {
                        row0 = xc+R;
                        col = col0;
                        goto srch_vt_dir;
                }
/* theta near 0 degree */
                if ( theta<0.008726 && theta>0 )/*0.5degree,ie Pi/360 */
                {
                        row0 = xc+R;
                        col = col0;
                        goto srch_vt_dir;
                }
/* angles other than Pi and 2*Pi */
                row0 = xc + (int)( R*cos(theta)); /* assumed arc */
                if ( row0<brow_st-4 || row0>brow_end+2 )
                continue;
                G_line(0,row0,col0,xc,yc,GREEN);
                goto normal_angle;
srch_vt_dir:
                G_line(0,row0,col0,xc,yc,GREEN);
                for( k=0;row0+k<brow_end+2; k++)
                {
                        if ( row0-k<brow_st-4 )
                        break;
                        row=row0+k;
vt_p:
                        pix_val=Vwix(&fbu[2],row,col);
                        if ( pix_val==255 )
                        {
                                r=abs(row-xc);
                                if ( r>Rx )  Rx = r;
                        printf("\n theta=%f r=%.2f (r-R)/dd=%f",theta,r,(r-R)/dd);
                                if ( (fabs( r - R))/dd <Rratio )
                                break;
                                if ( xc > 0 && xc<512 && yc>0 && yc<512)
                                G_line(0,row,col,xc,yc,BLUE);
                                num_out_arc++;
                                break;
                        }
                        if ( row==row0-k )
                        continue;
                        else
```

```
                        row = row0 -k;
                        goto vt_p;
                }
                continue;
normal_angle:
                for ( i=0; col0+i<col_end;i++ )
                {
                        if ( col0-i < col_st )
                        break;
                        col = col0 +i;
arc_p:
                        if ( (theta<1.5*PI+0.008726) && (theta>1.5*PI-0.008726)
)/*0.5degree,ie Pi/360 */
                        row = xc;
                        else if ( (theta<PI/2+0.008726) && (theta>PI/2-0.008726)
)/*^ 5degree,ie Pi/360 */
                        row = xc;
                        else
                        row = xc + (int)((double)(col- yc)/tan(theta));
/*
                        if ( tan(theta)>0 )
                        row = xc + (int)sqrt( R*R - (double)((col-yc)*(col-yc)) );
                        else
                        row = xc - (int)sqrt( R*R - (double)((col-yc)*(col-yc)) );
*/
                        if ( row<brow_st-4 || row>brow_end+2 )
                        break;
                        if ( xc > 0 && xc<512 && yc>0 && yc<512)
                        G_line(0,row,col,row0,col0,RED);

                        pix_val = Vwix( &fbu[2],row,col);
                        if ( pix_val == 255 )
                        {
                        /*      if ( xc > 0 && xc<512 && yc>0 && yc<512)
                                G_line(0,row,col,xc,yc,GREEN);*/
                                r = fabs((double)(col-yc)/sin(theta));
                                if ( r>Rx )  Rx = r;
                                printf("\n theta=%f r=%.2f (r-R)/dd=%f",theta,r,(r-R)/dd);
                        /*      if ( (fabs( r - R))/R <Rratio )         */
                                if ( (fabs( r - R))/dd <Rratio )
                                break;
                                if ( xc > 0 && xc<512 && yc>0 && yc<512)
                                G_line(0,row,col,xc,yc,BLUE);
                /*              printf("\n theta=%f r=%.2f (r-R)/R=%f",theta,r,(r-R)/R); */
                                num_out_arc++;
                                break;
                        }
                        if ( col==col0-i )
                        continue;
                        else
                        col = col0 -i;
                        goto arc_p;
                }
        }
        fit_ratio = fabs((Rx-R)/dd);
```

```
        printf("\n %d  points out of the presumed arc,fit_ratio=%f",num_out_arc,fit_ratio);
        if ( num_out_arc < Num_out_arc ) {
        flg_arc=1;
        printf("\n It is a ARC. ");
        } else
        printf("\n It is NOT a ARC. ");
        printf("\nr1=%f r2=%f rm=%f",r1,r2,rm);
        printf("\n ARC: R=%f c(%d,%d) \n R_p=%f R0=%f
pc(%d,%d)",R,xc,yc,R_p,R0,pp[3].c_row,pp[3].c_col);
        printf("\ntheta1=%f theta2=%f",theta1,theta2);
        if ( yt2< yt1 )
        {
                x1 = xt2;
                y1 = yt2;
                x2 = xt1;
                y2 = yt1;
        }
        return;
}
#define  NK     10
#define         Dh      6
line_detection()
{
        register  en;

        int     i,j,kk,pix_val,row,col,row_st,row_end,row0,col0,on_num=0;
        int     r1,c1,r2,c2,m,i_xh,flg_ge,brk_no[256];
        double  dh[256],ddh;
        coor_t  dhxy[256];

/* Check small segments */
        flg_p=0;
        flg_mh=0;
        Vdispl(&fbu[2],0,0,0);
        G_line(0,x1,y1,x2,y2,RED);
        if ( abs(y2-y1) <KK && abs(x2-x1) <KK )
        {       flg_p = 1;
                printf("\n IT is a Small Segment OR Parrell line OR vertical line.\n");
                return 0;
        }
        r1 = x1;
        c1 = y1;
        r2 = x2;
        c2 = y2;
        if ( y2 < y1 )
        {
                x1 = r2;
                y1 = c2;
                x2 = r1;
                y2 = c1;
        }
        if ( x2>x1 )
        {  row_st= x1-1;
           row_end= x2+1;
        }
```

```
        else
        {       row_st = x2-1;
                row_end = x1+1;
        }
/* compute afa,belta,vertical distance between Pa and the chord P1P2   */
        kxy = ((float)(x2 -x1))/((float)(y2-y1));
        afa = atan (kxy);
        dd=distance(x1,y1,x2,y2);
        brk_num = 0;
        flg_line=0;
        flg_ge = 0;
line:
/* Check the vertical distance */
        max_dh = 0;
        dh[0] = 0;
        if ( y2-y1 > NK ) {
                kk = (y2-y1)/NK;
                if ( kk >2 )
                kk = 2;
        } else
        kk = 1;
        m = (y2-y1)/kk/2;  /* middle point */
        for (i=0; i<2*m; i++ )
        dh[i] = 0;
        for( i=1;y1+i*kk<y2;i++)
        {
                col0 = y1 + i*kk;
                row0= x1 + (int)((float)(i*kk)*kxy);/* P1P2 */
                for(j=0;col0+j<y2;j++)
                {
                        col= col0+j;
line_hh:
                        row=row0+(int)((double)(col-col0)*tan(afa+PI/2.0)); /* VT Line */
                        if ( row <row_st || row > row_end )
                        continue;
                        pix_val = Vwix(&fbu[2],row,col);
                        if ( pix_val == 255  ){
                                on_num++;
                                dh[i]=fabs((double)(row-row0)/cos(afa));
                                dhxy[i].c_row = row;
                                dhxy[i].c_col = col;
                                G_line(0,row0,col0,row,col,GREEN);
                                printf("\n dh[%d]=%f dh/dd=%f",i,dh[i],dh[i]/dd);
                        if ( dh[i]/dd > Lratio ) {
                                brk_num++;
                                brk_no[brk_num] = i;
                                G_line(0,row0,col0,row,col,BLUE);
                        }
                        break;
                        }
                        if ( col <= col0 )
                        continue;
                        col=col0-j;
                        if ( col < y1 )
                        continue;
```

```
                    else
                    goto line_hh;
                }
        }
        if ( on_num==0)
        goto no_line;
        if ( brk_num < NUMbk )
        goto straight;
/* check isolated points produced by blank points */
        printf("\nCheck isolated points. Break points number: %d", brk_num);
        for( i=3;y1+i*kk<y2-2*kk;i++)
        {
                if ( dh[i]/dd >Lratio )
                {
                        printf("\n dh[%d]=%f dh/dd=%f",i,dh[i],dh[i]/dd);
                if ( dh[i-1]/dd< Lratio && dh[i+1]/dd<Lratio )
                {
                        for( j=1; j<2; j++ )
                        {
                                if ( dh[i-j]> 0 ) {
                                        dh[i]=dh[i-j];
                                        brk_num--;
                                        break;
                                }
                                if ( dh[i+j] >0 ) {
                                        dh[i]=dh[i+j];
                                        brk_num--;
                                        break;
                                }
                                dh[i] = dh[i-1];
                                brk_num--;
                        }
                }
                }
        }
/* check isolated blank value ( dh=0 ) points */
        printf("\n Check blank points. Break points number: %d",brk_num);
        for( i=3;y1+i*kk<y2-2*kk;i++)
        {
                if ( dh[i]/dd >Lratio )
                        printf("\n dh[%d]=%f dh/dd=%f",i,dh[i],dh[i]/dd);
        }
/*      for ( i=2; y1+j*kk<y2-2*kk; i++)
        {
                if ( dh[i] == 0 )
                {
                        for ( j=1; j<3; j++ )
                        {
                                if ( dh[i+j]>0 )
                                {
                                        dh[i]=dh[i+j];
                                        if ( dh[i]/dd> Lratio)
                                        brk_num++;
                                        break;
                                }
```

```
                                        if ( dh[i-j] >0 )
                                        {
                                                dh[i]=dh[i-j];
                                                if ( dh[i]/dd> Lratio)
                                                brk_num++;
                                                break;
                                        }
                                }

        }
*/
/* Search for Max_h point*/
        max_dh = 0;
        for( i=1;y1+i*kk<y2;i++)
        {
                if ( dh[i] > max_dh ) {
                max_dh = dh[i];
                i_xh = i;
                maxh.c_row = dhxy[i].c_row;
                maxh.c_col = dhxy[i].c_col;
                }
        }

        if ( brk_num <NUMbk )
        goto straight;
        else {

        printf("\nThere are %d points outside threshold,on_num=%d",brk_num,on_num);
        printf("\nmax_vd=%f at(%d,%d) i_xh=%d mid=%d
dh[m]=%f",max_dh,maxh.c_row,maxh.c_col,i_xh,m,dh[m]);
        G_line(0,x1,y1,maxh.c_row,maxh.c_col,GREEN);
        /* check if the dh of the mid point is around the point of Max_dh */
        if ( i_xh >m ) {
                if ( i_xh == m+1 ) {
                        flg_mh = 0;
                        goto no_line;
                }
                for ( i=0; i<3; i++ )
                {
                        ddh = fabs(dh[m+i]-max_dh);
                        if( (ddh/dd<0.01) && ddh<1 )
                        {
                                flg_mh = 0;
                                goto no_line;
                        }else {
                                flg_mh = 1;
                                goto no_line;
                        }
                }
        }
        if ( i_xh <m ) {
                if ( i_xh == m-1 ) {
                        flg_mh = 0;
                        goto no_line;
                }
```

```
                for ( i=0; i<3; i++ )
                {
                        ddh = fabs(dh[m-i]-max_dh);
                        if( (ddh/dd<0.01) && ddh<1 )
                        {
                                flg_mh = 0;
                                goto no_line;
                        }else {
                                flg_mh = 1;
                                goto no_line;
                        }
                }
        }
        if ( i_xh ==m )
        flg_mh = 0;
no_line:
        printf("\n It is NOT a LINE. flg_mh=%d",flg_mh);
        goto line_back;
        }
straight:
        flg_line=1;
printf("\n It is a straight LINE. \n");
line_bac'
printf("\n\nere are %d points outside threshold,on_num=%d",brk_num,on_num);
printf("\nmax_vd=%f at(%d,%d) i_xh=%d mid=%d dh[m]=%f
",max_dh,maxh.c_row,maxh.c_col,i_xh,m,dh[m]);
        fit_ratio = max_dh/dd;
printf("\n fit_ratio=%f", fit_ratio);
        x1 = r1;
        y1 = c1;
        x2
        y2 = c2;
        printf("\nEnd of line detection,click to return.");
        for (en = 0;;)
                if (Gcursor(2))
                        if (Gcursor(3,0) == 0) en = 1;
                        else if (en) break;
        printf("\n");
        return 1·
}

search_max()
{

        int     i,j,kk,pix_val,row,col,row_st,row_end,row0,col0;
        int     r1,c1,r2,c2,m,i_xh;
        double  dh[256],ddh,vh;
        coor_t  dhxy[256];
/* compute afa,delta,vertical distance between Pa and the chord P1P2   */
        kxy = ((float)(x2 -x1))/((float)(y2-y1));
        afa = atan (kxy);
        dd=distance(x1,y1,x2,y2);
/* Check the vertical distance */
        max_dh = 0;
        dh[0] = 0;
```

```
        if ( y2-y1 > NK ) {
                kk = (y2-y1)/NK;
                if ( kk >2 )
                kk = 2;
        } else
        kk = 1;
        for( i=1;y1+i*kk<y2;i++)
        {
                col0 = y1 + i*kk;
                row0= x1 + (int)((float)(i*kk)*kxy);/* P1P2 */
                dh[i] = 0;
                for(j=0;col0+j<y2;j++)
                {
                        col= col0+j;
line_hh:
                        row=row0+(int)((double)(col-col0)*tan(afa+PI/2.0)); /* VT Line */
                        if ( row <row_st || row > row_end )
                        continue;
                        pix_val = Vwtx(&fbuf[2],row,col);
                        if ( pix_val == 255 ){
                                vh=fabs((double)(row-row0)/cos(afa));
                                dhxy[i].c_row = row;
                                dhxy[i].c_col = col;
                                G_line(0,row0,col0,row,col,GREEN);
                        if ( vh > dh[i] ) {
                                dh[i] = vh;
                                G_line(0,row0,col0,row,col,BLUE);
                        }
                        }
                        if ( col <= col0 )
                        continue;
                        col=col0-j;
                        if ( col < y1 )
                        continue
                        else
                        goto line_hh;
                }
        }

/* Search for Max_h point*/
        max_dh = 0;
        for( i=1;y1+i*kk<y2;i++)
        {
                if ( dh[i] >= max_dh ) {
                max_dh = dh[i];
                i_xh = i;
                maxh.c_row = dhxy[i].c_row;
                maxh.c_col = dhxy[i].c_col;
                }
        }
        return;
}

#define  SMlen   6
```

```
primitive_chk( int n )
{
        line_detection();
        if (  flg_p == 1)
        {
                pr[n].type = 1;
                pr[n].len = distance(x1,y1,x2,y2);
                pr[n].bta = angle_compute(x1,y1,x2,y2);
        /*      if ( pr[n].len >SMlen )
                pr[n].type = 2; */
                return;
        }
        if ( flg_line == 1 )
        {
                pr[n].type = 2;
                pr[n].len = distance(x1,y1,x2,y2);
                pr[n].bta = angle_compute(x1,y1,x2,y2);
                return;
        }
        if ( flg_mh ==1 )
        goto pr_4;
        arc_detection();
        if ( flg_arc == 1 )
        {
                pr[n].type = 3;
                return;
        }
pr_4:
        pr[n].type = 4;
        return;
}

blob_threshold()
{
/* Threshold the image */
        Vmkmap(1,2,128);
        Vcopy(&fbu[3],1,&fbu[1]);
        Vdispl(&fbu[1],0,0,0);
        return;
}

b_histogram(flg,b)
        but_t    *b;
{
        register  en, key;
        coor_t              svpos;
        win_t    Whist;
        int      histogram[256],dbuf[256];
        int      max_value[2], max_index[2],i;
        int      p1,p2,auto_thres;
        int      max;

        WI(&Whist, 256,10,256,256 );     /*      histogram window */
        switch (flg) {
        case F_RFSH: showbut(b,BUBCOL,BUTCOL); break;
```

```
        case F_EXEC:
                Gcursor(0);
                Gcursor(3,&svpos);
                showbut(b,BUHCOL,BUTCOL);

        max = (512*512)/100;
        Vdispl(&fbu[3],0,0,0);
        Vhist( &fbu[3], 0, histogram );
                printf(" display histogram,click to continue ");
                for (en = 0;;)
                        if (Gcursor(2))
                                if (Gcursor(3,0) == 0) en = 1;
                                else if (en) break;
                printf(" \nBar graph, click to continue ");
        Vfill(&Whist,0);
        Gb_fit(histogram,256,dbuf,Whist.w_ncol);
        Gb_scale(dbuf,Whist.w_ncol,Whist.w_nrow,max);
        Gplot( 1, &Whist, dbuf,0,255,1);
        Vdispl(&fbu[0],0,0,0);
/* find max peak */
        for ( max_value[1] = max_index[1] =0, i=0, i<256; i++) {
                if ( histogram[i] > max_value[1] ) {
                        max_value[1] = histogram[i];
                        max_index[1] = i;
                }
        }
        printf("\n The peak has %d units, at grey level %d",max_value[1],max_index[1]);


/* auto threshold */
        printf("\n Auto threshold,click to continue ");
        for (en = 0;;)
                if (Gcursor(2))
                        if (Gcursor(3,0) == 0) en = 1;
                        else if (en) break;
        printf("\n");
        p1 = 60;
        p2 = 0;
        auto_thres = Vhthres(0,histogram,p1,p2);
        printf("\n  p1=%d, p2=%d, auto thresold value =%d",p1,p2,auto_thres );
        if ( auto_thres >255 || auto_thres <0 ) {
                printf("\n Auto threshold fail. ");
                goto exit;
        }
        Vmkmap(1,2,auto_thres);
        Vcopy(&fbu[3],1,&fbu[1]);
        Vdispl(&fbu[1],0,0,0);
exit:
        printf("\n Click to return to main menu. ");
        for (en = 0;;)
                if (Gcursor(2))
                        if (Gcursor(3,0) == 0) en = 1;
                        else if (en) break;
                Vdispl(&fbu[0],0,0,0);
                showbut(b,BUBCOL,BUTCOL);
```

```
                Gcursor(4,&svpos);
                Gcursor(1);
                break;
        }
}


#define         Min_edge        10
#define         Min_area        20

blob_label()
{
        int     i,j,k,bkground;

        Vblob(&fbu[1],0,&fbu[1],&Nblack,&Nwhite);
        printf("Nblack = %d Nwhite = %d\n",Nblack,Nwhite);
        if (Nblack <= 128 && Nwhite <= 128)
        {
                Vbarea(&fbu[1],barea);
                Vbbox(&fbu[1],bbox);
                /* eliminate small area <15 */
                for (i=0,j=0; j<Nblack; ++j,i+=2)
                {
                        printf("area[%d] = %6d\n",i,barea[i]);
                        if (barea[i] < Min_area ) {
                                bbox[i].w_ncol = 0;
                        }
                }
                for (i=1,j=0; j<Nwhite; ++j,i+=2)
                {
                        printf("area[%d] = %6d\n",i,barea[i]);
                        if (barea[i] < Min_area ) {
                                bbox[i].w_ncol = 0;
                        }
                }
                /* draw the bounding box */
                for (i=0, j=0; j< Nblack; ++j,i+=2)
                {
                        if (bbox[i].w_ncol != 0)
                                G_rect(0,&bbox[i],RED,0);
                }
                for (i=1, j=0; j< Nblack; ++j,i+=2)
                {
                        if (bbox[i].w_ncol != 0)
                                G_rect(0,&bbox[i],GREEN,0);
                }
        } else {
                printf(" Too many blobs found\n");
                return 0;
                }

/* Building blob tree relationship */

        bkground = Vbbtree(bbox, Nblack, Nwhite, Min_edge, btree );
        for ( k=2,j=1; j<Nblack; j++, k+=2 )
```

```
        {
                for ( i=0; i<4; i++ )
                printf("\n  blob[%d][%d]  %d ",k,i,btree[k][i]);
        }
        printf("\n Backgound colour: %d",bkground);
        return 1;
}
int     P8, P4;

int blob_perimeter(int row_st, int col_st, int row_end, int col_end )
{
        int     i,j,k,l, a[8];
        int     perimeter;

        P8 = 0;
        P4 = 0;
        for ( i=row_st; i<row_end; i++ )
        {
        for ( j=col_st; j<col_end; j++ )
        {
                a[0] = Vwix( &fbu[2],i,j);
                if ( a[0] ==0 )
                continue;
                if ( a[0] == 255 ) {
                        a[1] = Vwix( &fbu[2],i-1,j-1);
                        a[2] = Vwix( &fbu[2],i-1,j);
                        a[3] = Vwix( &fbu[2],i-1,j+1);
                        a[4] = Vwix( &fbu[2],i,j+1);
                        a[5] = Vwix( &fbu[2],i+1,j+1);
                        a[6] = Vwix( &fbu[2],i+1,j);
                        a[7] = Vwix( &fbu[2],i+1,j-1);
                        a[8] = Vwix( &fbu[2],i,j-1);
                        for ( k=1; k<9; k++ ) {
                                if ( a[k]==0 ) {
                                P8++;
                                break;
                                }
                        }
                        if ( a[2]==0 || a[4]==0 || a[6]==0 || a[8]==0 )
                                P4++;
                }
        }
        }
        perimeter = (int)( sqrt( (double)(P8*P4) ) );
        printf("\n St(%d,%d) End(%d,%d)",row_st,col_st,row_end, col_end);
        printf("\n P4=%d P8=%d Peri=%d", P4, P8, perimeter);
        return ( perimeter);
}

blob_compute( int blob_no )
{
        unsigned int    bmomt1[3],blob_area;
        int             B_lut,r1,t2, blob_peri;
        float   blob_compact,blob_thin;
```

```
        brow_st=bbox[blob_no].w_row;
        brow_end=bbox[blob_no].w_nrow+brow_st;
        bcol_st=bbox[blob_no].w_col;
        bcol_end=bbox[blob_no].w_ncol+bcol_st;
/*
        Vmkmap(blob_no,11,blob_no);
        Vcopy(&fbu[1],blob_no,&fbu[2]);
        Vmomt1(&fbu[1],blob_no,bmomt1);
*/
        Vmkmap(1,1);    /* reverts image */
        B_lut = 6;
        t1 = blob_no ;
        t2 = btree[blob_no][2];                 /* leftmost child of blob_no */
        printf("\n t1 =%d t2 = %d",t1,t2);
        if ( t2 ==-1 ) {
                Vmkmap(B_lut,11,blob_no);
                Vcopy(&fbu[1],B_lut,&fbu[2]);
        } else {
                Vmkmap(B_lut,5,t1,t2+1 );
                Vcopy(&fbu[1],B_lut,&fbu[2]);
                Vcopy(&fbu[2],1,&fbu[2]);
        }
        Vdispl(&fbu[2],0,0,0);
        Vmomt1(&fbu[2],0,bmomt1);

        blob_area = bmomt1[0]/255;
        blob[blob_no].area = blob_area;
        printf("blob area=%d",blob_area);
/* perimeter computation */
        printf("\n Perimeter computation. Waiting...");
        printf("\n Waiting...");
        blob_peri = blob_perimeter(brow_st+5,bcol_st+5,brow_end+5,bcol_end+5);
        printf("\nPerimeter=%d",blob_peri);
        blob[blob_no].peri = blob_peri;
        blob_compact =(float)( (float)(P8*P4)/((float)(blob_area)) );
        blob_thin = 4*PI/blob_compact;
        blob[blob_no].compct = blob_compact;
        blob[blob_no].thin = blob_thin;
        printf("\nCompactness ratio=%.3f  thinness ratio=%.3f",blob_compact,blob_thin);

        if ( bmomt1[0] >0 )
        {
        xo = abs( bmomt1[1]/bmomt1[0]);
        yo = abs( bmomt1[2]/bmomt1[0]);
        blob[blob_no].xo = xo;
        blob[blob_no].yo = yo;
        printf("\n momnt[1]=%d momnt[2]=%d",bmomt1[1]/255,bmomt1[2]/255);
        printf("\t Gravity Center xo = %d  yo = %d ", xo,yo);
        printf("\n");
        } else{
                printf("Area<= 0 overflow!");
                return 0;
                }
        Vdispl(&fbu[2],0,0,0);
        GHline(0,xo,yo,4,0xc5);
```

```
        GHline(0,xo,yo,-4,0xc5);
        GVline(0,xo,yo,-4,0xc5);
        GVline(0,xo,yo,4,0xc5);

        return 1;
}

blob_edge( int blob_no )
{
/* outline of the blob */
        VClapla5(&fbu[2],0,&fbu[2]);
        Vdispl(&fbu[2],0,0,0);
        G_rect(0,&bbox[blob_no],0xc5,0);
        return;
}

b_edge_compare(flg,b)
        but_t    *b;
{
        register  en, key;
        coor_t              svpos;

        switch (flg) {
        case F_RFSH: showbut(b,BUBCOL,BUTCOL); break;
        case F_EXEC:
                Gcursor(0);
                Gcursor(3,&svpos);
                showbut(b,BUHCOL,BUTCOL);

        printf("\n Horizontal Emphasis.");
        VHemph(&fbu[3],0,&fbu[2]);
        Vdispl(&fbu[2],0,0,0);

        printf("\n Vertical Emphasis, click to continue.");
        for (en = 0;;)
        if (Gcursor(2))
                if (Gcursor(3,0) == 0) en = 1;
                        else if (en) break;
                printf("\n");
        VVemph(&fbu[3],0,&fbu[2]);
        Vdispl(&fbu[2],0,0,0);

        printf("\n Laplacian Cross3x3, click to continue.");
        for (en = 0;;)
        if (Gcursor(2))
                if (Gcursor(3,0) == 0) en = 1;
                        else if (en) break;
                printf("\n");
        VClapla3(&fbu[3],0,&fbu[2]);
        Vdispl(&fbu[2],0,0,0);

        printf("\n Laplacian Cross5x5, click to continue.");
        for (en = 0;;)
        if (Gcursor(2))
                if (Gcursor(3,0) == 0) en = 1;
```

```
                else if (en) break;
            printf("\n");
        VClapla5(&fbu[3],0,&fbu[2]);
        Vdispl(&fbu[2],0,0,0);

        printf("\n Laplacian Cross7x7, click to continue.");
        for (en = 0;;)
        if (Gcursor(2))
            if (Gcursor(3,0) == 0) en = 1;
                else if (en) break;
            printf("\n");
        VClapla7(&fbu[3],0,&fbu[2]);
        Vdispl(&fbu[2],0,0,0);

        printf("\n Click to exit.");
        for (en = 0;;)
        if (Gcursor(2))
            if (Gcursor(3,0) == 0) en = 1;
                else if (en) break;
            printf("\n");

            Vdispl(&fbu[0],0,0,0);
            showbut(b,BUBCOL,BUTCOL);
            Gcursor(4,&svpos);
            Gcursor(1);
            break;
        }
        return;
}
/*****************************************
        Fuction:Primitive verification
        INPUT: cp[1]--cp[Ncp]
        OUTPUT:     pr[1]--pr[Ncp]
            Ncp
*****************************************/
primitive_extract()
{
        register  en;
        int             i,k;

        for ( i=1; i<Ncp; i++ )
        {
chk_prim:
            y1 = cp[i].c_col;
            x1 = cp[i].c_row;
            y2 = cp[i+1].c_col;
            x2 = cp[i+1].c_row;
            if ( y2 < y1 )
            flg_dir = 1;
            else
            flg_dir = 0;
            line_detection();
            if ( flg_p == 1)
            {
                pr[i].type = 1;
```

```
                        pr[i].len = distance(x1,y1,x2,y2);
                        pr[i].bta = angle_compute(x1,y1,x2,y2);
/*                      if ( pr[i].len >SMlen )
                        pr[i].type = 2;*/
                        goto next_prim;
                }
                if ( flg_line == 1 )
                {
                        pr[i].type = 2;
                        pr[i].len = distance(x1,y1,x2,y2);
                        pr[i].bta = angle_compute(x1,y1,x2,y2);
                        goto next_prim;
                }
                if ( flg_mh ==1 )
                {       pr[i].type = 4;
                        goto tp_4;
                }
                arc_detection();
                if ( flg_arc == 1 )
                {
                        pr[i].type = 3;
                        goto next_prim;
                }
tp_4:
                flg_brk++;
                if ( flg_brk>1 )
                goto next_prim;
                for ( k=Ncp; k>i; k-- )
                {
                        cp[k+1].c_row = cp[k].c_row;
                        cp[k+1].c_col = cp[k].c_col;
                }
                cp[i+1].c_row = maxh.c_row;
                cp[i+1].c_col = maxh.c_col;
                Ncp++;
                goto chk_prim;
next_prim:
        printf("\nAnother primitive detection (i=%d),click to continue ",i);
        for (en = 0;;)
        if (Gcursor(2))
                if (Gcursor(3,0) == 0) en = 1;
                        else if (en) break;
                printf("\n");

        flg_brk = 0;
        continue;
        }
/* check the last primitive */
        flg_dir = 1;
        x1 = cp[Ncp].c_row;
        y1 = cp[Ncp].c_col;
        x2 = cp[1].c_row;
        y2 = cp[1].c_col;
        primitive_chk(Ncp);
        if ( pr[Ncp].type >3 )
```

```
        {
                line_detection();
                cp[Ncp+1].c_row = maxh.c_row;
                cp[Ncp+1].c_col = maxh.c_col;
                x1 = cp[Ncp].c_row;
                y1 = cp[Ncp].c_col;
                x2 = cp[Ncp+1].c_row;
                y2 = cp[Ncp+1].c_col;
                primitive_chk(Ncp);
/*              if ( pr[Ncp].type >3 )
                {
                        Ncp++;
                        return;
                } */
                x1 = cp[Ncp+1].c_row;
                y1 = cp[Ncp+1].c_col;
                x2 = cp[1].c_row;
                y2 = cp[1].c_col;
                primitive_chk( Ncp+1 );
                Ncp++;
                return;
        }
}

/***********************************************
        FunctionMerge small segments
        Input           primitives pr[i]
        Output          modified pr[i], Ncp
***********************************************/

int     flg_for,flg_bck;
float   fit_for, fit_bck;

int small_seg_merge()
{
        int             i,j;
        Primitive       *pt1,*pt2;

        for ( i=1; i<Ncp+1; i++ ) {
                if ( pr[i].type == 1) {
/** merge verify **/
                merge_forward(i);
                merge_backward(i);
/** merge decision **/
                if ( flg_for==0 && flg_bck==0 )
                        continue;
                if ( flg_for>0 && flg_bck>0 ) {
                        if ( fit_for > fit_bck ) {
                                pr[i].type = flg_bck;
                                adjust_cp(i-1);
                        } else {
                                pr[i].type = flg_for;
                                adjust_cp(i);
                        }
                } else {
```

```
                        if ( flg_for>0 ) {
                                pr[i].type = flg_for;
                                adjust_cp(i);
                        }
                        if ( flg_bck >0 ) {
                                pr[i].type = flg_bck;
                                adjust_cp(i-1);
                        }
                }
                }
        }
        return(0);

}

int re_merge( )
{
        int     i;
        double  left_l,right_l;

        flg_for =0;
        flg_bck=0;
        if ( Dt_lnum == 0 ) {
        for ( i=1; i<Ncp+1; i++ ) {
                if ( pr[i].type == 3 ) {
                        if ( i==1 )
                                left_l = pr[Ncp].len;
                        else if ( i == Ncp )
                                right_l = pr[1].len;
                                else {
                                        left_l = pr[i-1].len;
                                        right_l = pr[i+1].len;
                                }
                        if ( left_l >right_l )
                                break_back(i);
                        else
                                break_for(i);
                }
        }
        }

        if ( Dt_lnum >0 ) {
        for ( i=1; i<Ncp+1; i++ ) {
                if ( pr[i].type == 3 ) {
                        if ( i==1 )
                                left_l = pr[Ncp].len;
                        else if ( i == Ncp )
                                right_l = pr[1].len;
                        if ( i>1 && i<Ncp ) {
                                left_l = pr[i-1].len;
                                right_l = pr[i+1].len;
                        }
                        if ( left_l >right_l )
                                merge_forward(i);
                        else
```

```
                                        merge_backward(i);


                                if ( flg_for>0 ) {
                                        pr[i].type = flg_for;
                                        adjust_cp(i);
                                }
                                if ( flg_bck >0 ) {
                                        pr[i].type = flg_bck;
                                        adjust_cp(i-1);
                                }
                        }
                }
                }
                return;
}


..t break_back(i)

        x1 = pr[i-1].P1.c_row;
        y1 = pr[i-1].P1.c_col;
        x2 = pr[i-1].P2.c_row;
        y2 = pr[i-1].P2.c_col;
        line_detection();
        x1 = maxh.c_row;
        y1 = maxh.c_col;
        x2 = pr[i].P2.c_row;
        y2 = pr[i].P2.c_col;
        arc_detection;
        if ( flg_arc== 1) {
                pr[i-1].P2.c_row = maxb.c_row;
                pr[i-1].P2.c_col = maxh.c_col;
                pr[i].P1.c_row = maxh.c_row;
                pr[i].P1.c_col = maxh.c_col;
                return 1;
        } else
                return 0;
}

int break_for(i)
{
        x1 = pr[i+1].P1.c_row;
        y1 = pr[i+1].P1.c_col;
        x2 = pr[i+1].P2.c_row;
        y2 = pr[i+1].P2.c_col;
        line_detection();
        x2 = maxh.c_row;
        y2 = maxh.c_col;
        x1 = pr[i].P1.c_row;
        y1 = pr[i].P1.c_col;
        arc_detection;
        if ( flg_arc== 1) {
                pr[i+1].P1.c_row = maxh.c_row;
                pr[i+1].P1.c_col = maxh.c_col;
```

```
                pr[i].P2.c_row = maxh.c_row;
                pr[i].P2.c_col = maxh.c_col;
                return 1;
        } else
                return 0;
}


int merge_forward(int n )
{
        Primitive       *pt1,*pt2;
/*** merge with the right ***/
        if ( n= Ncp )
                pt2 = &pr[1];
        else
                pt2 = &pr[n+1];
        pt1 = &pr[n];
        x1 = pt1->P1.c_row;
        y1 = pt1->P1.c_col;
        x2 = pt2->P2.c_row;
        y2 = pt2->P2.c_col;
        if ( pt1->type == 3 ) {
                arc_detection();
                if ( flg_arc == 1) {
                        flg_for = 3;
                        fit_for = fit_ratio;
                } else
                        flg_for = 0;
        }
        else {
                line_detection();
                if ( flg_line == 1) {
                        flg_for = 2;
                        fit_for = fit_ratio;
                } else {
                        arc_detection();
                        if ( flg_arc == 1 ) {
                                flg_for =3;
                                fit_for = fit_ratio;
                        } else
                                flg_for = 0;
                }
        }
        return;
}

int merge_backward( int n)
{
        Primitive       *pt1,*pt2;
/*** merge with the left ***/
        if ( n= Ncp )
                pt2 = &pr[1];
        else
                pt2 = &pr[n+1];
        pt1 = &pr[n];
```

```
        x1 = pt1->P1.c_row;
        y1 = pt1->P1.c_col;
        x2 = pt2->P2.c_row;
        y2 = pt2->P2.c_col;
        if ( pt1->type == 3 ) {
                arc_detection();
                if ( flg_arc == 1) {
                        flg_bck = 3;
                        fit_bck = fit_ratio;
                } else
                        flg_bck = 0;
        }
        else {
                line_detection();
                if ( flg_line == 1) {
                        flg_bck = 2;
                        fit_bck = fit_ratio;
                } else {
                        arc_detection();
                        if ( flg_arc == 1 ) {
                                flg_for =3;
                                fit_for = fit_ratio;
                        } else
                                flg_bck = 0;
                }
        }
        return;
}

int adjust_cp( int n )
{
        int j;
        Primitive       *pt1,*pt2;

        num_merge++;
        if ( n==0 || n==Ncp ) {
                pt1 = &pr[1];
                pr[1].P1.c_row = pr[Ncp].P1.c_row;
                pr[1].P1.c_col = pr[Ncp].P1.c_col;
                x1 = pt1->P1.c_row;
                y1 = pt1->P1.c_col;
                x2 = pt1->P2.c_row;
                y2 = pt1->P2.c_col;
                pr[1].len = distance(x1,y1,x2,y2);
                pr[1].bta = angle_compute(x1,y1,x2,y2);
                Ncp--;
                return;
        }


        pt1 = &pr[n];
        pr[n].P2.c_row = pr[n+1].P2.c_row;
        pr[n].P2.c_col = pr[n+1].P2.c_col;
        x1 = pt1->P1.c_row;
        y1 = pt1->P1.c_col;
```

```
        x2 = pt1->P2.c_row;
        y2 = pt1->P2.c_col;
        pr[n].len = distance(x1,y1,x2,y2);
        pr[n].bta = angle_compute(x1,y1,x2,y2);
        for ( j=n+1; j<Ncp; j++ )
        {
                pr[j].P1.c_row = pr[j+1].P1.c_row;
                pr[j].P1.c_col = pr[j+1].P1.c_col;
                pr[j].P2.c_row = pr[j+1].P2.c_row;
                pr[j].P2.c_col = pr[j+1].P2.c_col;
                pr[j].type = pr[j+1].type;
                pr[j].len = pr[j+1].len;
                pr[j].bta = pr[j+1].bta;
                pr[j].sym = pr[j+1].sym;
        }
        Ncp--;
        return;
}


/*************************************** ** ********
        Fuction         Merge lines
        Input           primitives pr[i]
        Output          modified pr[i], Ncp
*************************** ***********************/
int line_merge()
{
        int             i,j;
        double          dbta;
        Primitive       *pt1,*pt2;

        for ( i=1; i<Ncp; i++ )
        {
                if ( pr[i].type == 2 && pr[i+1].type==2 )
                {
                        dbta =  fabs( pr[i].bta-pr[i+1].bta );
                        printf("\n pr[%d].bta=%f pr[%d].bta=%f
delt=%f",i,pr[i].bta,i+1,pr[i+1].bta,dbta);
                        if ( dbta < Delt_angle )
                                goto merge_verify;
                        else if ( dbta<PI*2.0+Delt_angle && dbta>PI*2.0-Delt_angle )
                                goto merge_verify;
                        else
                                continue;
/** merge verify**/
merge_verify:
                        pt1 = &pr[i];
                        pt2 = &pr[i+1];
                        x1 = pt1->P1.c_row;
                        y1 = pt1->P1.c_col;
                        x2 = pt2->P2.c_row;
                        y2 = pt2->P2.c_col;
                        line_detection();
                        if ( flg_line ==1 ) {
                                num_merge++;
```

```
                        /* merge and adjust corner points*/
                                pr[i].type = 2;
                                pt1->P2.c_row = x2;
                                pt1->P2.c_col = y2;
                                pt1->len = distance(x1,y1,x2,y2);
                                pt1->bta = angle_compute(x1,y1,x2,y2);
                                for ( j=i+1; j<Ncp; j++ )
                                {
                                        pr[j].P1.c_row = pr[j+1].P1.c_row;
                                        pr[j].P1.c_col = pr[j+1].P1.c_col;
                                        pr[j].P2.c_row = pr[j+1].P2.c_row;
                                        pr[j].P2.c_col = pr[j+1].P2.c_col;
                                        pr[j].type = pr[j+1].type;
                                        pr[j].len = pr[j+1].len;
                                        pr[j].bta = pr[j+1].bta;
                                        pr[j].sym = pr[j+1].sym;
                                }
                                Ncp--;
                                continue;
                        }
                }
        }
        return;
}

/*********************************************
        Function: Merge arcs
        Input:  pr[Ncp], Ncp
        Output: pr[Ncp], Ncp
*********************************************/
arc_merge()
{
        Primitive       *pt1,*pt2;
        int     k,i,n_arc;

        for ( i=1; i<Ncp; i++ )
        {
                if ( pr[i].type==3 && pr[i+1].type>=3 )
                {
                        pt1 = &pr[i];
                        pt2 = &pr[i+1];
                        x1 = pt1->P1.c_row;
                        y1 = pt1->P1.c_col;
                        x2 = pt2->P2.c_row;
                        y2 = pt2->P2.c_col;
                        arc_detection();
                        if ( flg_arc==1 ) {
                                adjust_cp(i);
                                continue;
                        } else
                                continue;
                }
        }
/* connection between the first and the last two primitives */
        if ( pr[1].type>=3 && pr[Ncp].type==3 )
```

```
        {
                x1 = pr[Ncp].P1.c_row;
                y1 = pr[Ncp].P1.c_col;
                x2 = pr[1].P2.c_row;
                y2 = pr[1].P2.c_col;
                arc_detection();
                if ( flg_arc==1 )
                        adjust_cp( Ncp );
        }
}

io_primitive(int k)
{
        int     i;
        switch(k) {
        case 1:
        printf("\n Blob consists of ( %d ) primitives.",Ncp);
        printf("\n Primitive[n]      P1\t\t P2\t Type\t Length\t\t Angle");
        for ( i=1; i<Ncp+1; i++ )
        {
        printf("\n primitive[%d]\t(%d,%d)",i,pr[i].P1.c_row,pr[i].P1.c_col);
        printf(" (%d,%d)",pr[i].P2.c_row,pr[i].P2.c_col);
        printf("\t %d\t%f\t %f",pr[i].type,pr[i].len,pr[i].bta);
        }
        break;

        case 2:
        fprintf(io,"\n Blob consists of ( %d ) primitives.",Ncp);
        fprintf(io,"\n Primitive[n]   P1\t\t P2\t Type\t Length\t\t Angle");
        for ( i=1; i<Ncp+1; i++ )
        {
        fprintf(io,"\n primitive[%d]\t(%d,%d)",i,pr[i].P1.c_row,pr[i].P1.c_col);
        fprintf(io," (%d,%d)",pr[i].P2.c_row,pr[i].P2.c_col);
        fprintf(io,"\t %d\t%f\t %f",pr[i].type,pr[i].len,pr[i].bta);
        }
        break;
        }
        return;
}

blob_feature_extract( int n )
{
        int     i,j,num_len,k;
        double  dbta[10],dlen,dt;

        blob[n].line_num = 0;
        blob[n].arc_num = 0;
        blob[n].eql_num = 0;
        blob[n].eqa_num = 0;
        blob[n].eq_90 = 0;
        blob[n].gt_90 = 0;
        blob[n].les_90 = 0;

        if ( blob[n].circle == 1 )
                return;
```

```
for ( i=1; i<Ncp+'; i++ )
{
        dbta[i] = 0;
        if ( pr[i].type == 2 )
        {
                blob[n].line_num ++;
                if ( i<Ncp && pr[i+1].type==2 )
                        dbta[i] = fabs( PI-fabs( pr[i].bta - pr[i+1].bta ));
                if ( i==Ncp && pr[1].type==2 )
                        dbta[Ncp] = fabs( PI-fabs( pr[Ncp].bta - pr[1].bta ));
                        printf("\n delt_bta=%f",dbta[i]*180.0/PI);
                if ( dbta[i]>PI/2.0-Delt_angle && dbta[i]< PI/2.0+Delt_angle )
                        blob[n].eq_90 ++;
                if ( dbta[i]>= PI/2.0+ Delt_angle )
                        blob[n].gt_90 ++;
                if ( dbta[i]<= PI/2.0-Delt_angle )
                        blob[n].les_90 ++;
        }
        if ( pr[i].type == 3 )
                blob[n].arc_num ++;
}
/* compute the max number of identical lines ( equal length) */
for ( i=1; i<Ncp; i++ )
{
        num_len = 0;
        for ( j=i+1; j<Ncp+1; j++)
        {
                if ( pr[i].type == 2 && pr[j].type ==2 )
                {
                        dlen = fabs( pr[j].len - pr[i].len );
                        printf("\n delt_length=%f", dlen/pr[i].len);
                        if ( dlen/pr[i].len < Delt_len )
                        num_len++;
                }
        }
        if ( blob[n].eql_num< num_len )
                blob[n].eql_num = num_len;

}
if ( blob[n].eql_num>0 )
        blob[n].eql_num++;
for ( i=1; i<Ncp; i++ )
{
        k = 0;
        if ( dbta[i]>0 )
        {
                for ( j=i+1; j<Ncp+1; j++ )
                {
                        if ( dbta[j]>0 )
                        {
                                dt =   fabs(dbta[j] - dbta[i]);
                        printf("\n (%d,%d) dt_dbta=%f Degree=%f
Ratio=%f",i,j,dt,dt*180.0/PI, dt/dbta[i] );
```

```
                                        if( dt<Delt_angle || dt/dbta[i]< R_angle || dt/dbta[j]<
R_angle )
                                        k++;
                                }
                        }
                }
                if ( blob[n].eqa_num <k )
                        blob[n].eqa_num = k;
        }
        if ( blob[n].eqa_num >0 )
                blob[n].eqa_num++;

        return;
}

io_blob_feature( int k, int n )
{
        switch(k) {
        case 1:
        printf("\n Numbers of Lines: %d",blob[n].line_num);
        printf("\n Numbers of ARCs: %d",blob[n].arc_num);
        printf("\n Numbers of Lines with equal length: %d",blob[n].eql_num);
        printf("\n Numbers of equal angles: %d",blob[n].eqa_num);
        printf("\n Numbers of Angles =90 : %d",blob[n].eq_90);
        printf("\n Numbers of Angles >90 : %d",blob[n].gt_90);
        printf("\n Numbers of Angles <90 : %d",blob[n].les_90);
        printf("\n Area = %d", blob[n].area);
        printf("\n Gravity Center ( %d , %d )", blob[n].xo, blob[n].yo);
        printf("\n Perimeter = %d", blob[n].peri);
        printf("\n Compactness Ratio = %f", blob[n].compct);
        printf("\n Thinness Ratio = %f", blob[n].thin);
        break;

        case 2:
        fprintf(io,"\n Blob features are extracted as");
        fprintf(io,"\n Numbers of Lines: %d",blob[n].line_num);
        fprintf(io,"\n Numbers of ARCs: %d",blob[n].arc_num);
        fprintf(io,"\n Numbers of Lines with equal length: %d",blob[n].eql_num);
        fprintf(io,"\n Numbers of equal angles: %d",blob[n].eqa_num);
        fprintf(io,"\n Numbers of Angles =90 : %d",blob[n].eq_90);
        fprintf(io,"\n Numbers of Angles >90 : %d",blob[n].gt_90);
        fprintf(io,"\n Numbers of Angles <90 : %d",blob[n].les_90);
        fprintf(io,"\n Area = %d", blob[n].area);
        fprintf(io,"\n Gravity Center ( %d , %d )", blob[n].xo, blob[n].yo);
        fprintf(io,"\n Perimeter = %d", blob[n].peri);
        fprintf(io,"\n Compactness Ratio = %f", blob[n].compct);
        fprintf(io,"\n Thinness Ratio = %f", blob[n].thin);
        break;
        }
        return;
}

b_blob_analysis(flg,b)

        but_t    *b;
```

```
{
        register  en, key;
        coor_t          svpos;

        coor_t  *pt;
        int     i,j,k,n,pix_val,row,col,kk,bnum,seg_type[4];
        int     num_st,Num;

        switch (flg) {
        case F_RFSH: showbut(b,BUBCOL,BUTCOL); break;
        case F_EXEC:
                Gcursor(0);
                Gcursor(3,&svpos);
                showbut(b,BUHCOL,BUTCOL);

                blob_threshold();
/* Blob labelling in fbu[1] */
                blob_label();
/* Blob Analysis on Area,centroid,perimeter etc computation */
                printf(" Blob analysis,click to continue ");
                for (en = 0;;)
                        if (Gcursor(2))
                                if (Gcursor(3,0) == 0) en = 1;
                                else if (en) break;
                printf("\n");
/*blob singled out */
        num_st = 2;
        Num = Nblack;
blob_check:
        for ( k=num_st,n=1; n<Num; n++, k+=2 )
        {
        bnum=k;
        blob_compute( bnum );
        /* Edge finder */
        printf("\n blob no=%d",bnum);
                printf("\nLooking for edge and compute perimeter, click to continue ");
                for (en = 0;;)
                        if (Gcursor(2))
                                if (Gcursor(3,0) == 0) en = 1;
                                else if (en) break;
                printf("\n");
        blob_edge( bnum );

/* Find side point */
        printf("\n Search for side points...");
                for (en = 0;;)
                        if (Gcursor(2))
                                if (Gcursor(3,0) == 0) en = 1;
                                else if (en) break;
                printf("\n");
        side_point_detection(bnum);
        find_corner_point();
        printf("\n Ncp=%d",Ncp);
        for ( i=1; i<Ncp+1; i++ )
```

```
        {
        printf("\n cp[%d] (%d,%d)",i,cp[i].c_row,cp[i].c_col);
        G_line(0,cp[i].c_row,cp[i].c_col,xo,yo,RED);
        }
/*
        if ( Ncp == 2 ) {
inc_cp:         x1 = cp[1].c_row;
                y1 = cp[1].c_col;
                x2 = cp[2].c_row;
                   p[2].c_col;
                search_max();
                cp[3].c_row = maxh.c_row;
                cp[3].c_col = maxh.c_col;
                Ncp++;
        }

        for ( i=1; i<Ncp+1; i++ )
        {
        printf("\n cp[%d] (%d,%d)",i,cp[i].c_row,cp[i].c_col);
        G_line(0,cp[i].c_row,cp[i].c_col,xo,yo,RED);
        }
*/
        printf("\n blob no=%d",bnum);
printf("\n Circle detection,click to continue ");
                for (en = 0;;)
                        if (Gcursor(2))
                                if (Gcursor(3,0) == 0) en = 1;
                                else if (en) break;
                printf("\n");
        circle_detection( bnum );
        if ( flg_circle==0 ) {
                blob[bnum].circle = 0;
                goto prim_check;
        }
        if ( flg_circle==1 )
        {
                blob[bnum].circle=1;
                blob[bnum].name ="CIRCLE";
                goto blob_ex;
        }

prim_check:
        printf("\nPrimitive check,click to continue ");
                for (en = 0;;)
                        if (Gcursor(2))
                                if (Gcursor(3,0) == 0) en = 1;
                                else if (en) break;
                printf("\n");
        primitive_extract();
        for ( i=1; i<Ncp+1; i++ ){
        printf("\t cp[%d](%d,%d) pr=%d ",i,cp[i].c_row,cp[i].c_col, pr[i].type);
        pr[i].P1.c_row = cp[i].c_row;
        pr[i].P1.c_col = cp[i].c_col;
        if ( i==Ncp ) {
                pr[i].P2.c_row = cp[1].c_row;
```

```
                pr[i].P2.c_col = cp[1].c_col;
        } else {
                pr[i].P2.c_row = cp[i+1].c_row;
                pr[i].P2.c_col = cp[i+1].c_col;
        }
        }

/* display primitives */
        io_primitive(1);
        if ( *io_key == 'y' ) {
                printf("\nSave data. waiting...");
                io_primitive(2);
        }
merge_pr:
/* merge small segments */
        printf("\n Merge samll segments,  click to continue ");
        for (en = 0;;)
        if (Gcursor(2))
                if (Gcursor(3,0) == 0) en = 1;
                        else if (en) break;
                printf("\n");
        num_merge = 0;
        small_seg_merge();
        if ( num_merge >0 ) {
                io_primitive(1);
                if ( *io_key == 'y' ) {
                        fprintf(io,"\n After Small Segments merging:");
                        printf("\nSave data. waiting...");
                        io_primitive(2);
                }
        }
/* merge lines and small segments */
        printf("\n Merge lines,  click to continue ");
        for (en = 0;;)
        if (Gcursor(2))
                if (Gcursor(3,0) == 0) en = 1;
                        else if (en) break;
                printf("\n");
        num_merge = 0;
        line_merge();

        if ( num_merge >0 ) {
                io_primitive(1);
                if ( *io_key == 'y' ) {
                        fprintf(io,"\n After line merging:"),
                        printf("\nSave data. waiting...");
                        io_primitive(2);
                }
        }

/*      if ( Ncp ==2 )
                goto extend;
*/
/* connect adjacent arc */
connect_arc:
```

```
        printf("\nConn          tive, click to continue ");
        for (en = 0;;)
        if (Gcursor(2))
                if (Gcursor(3,0) == 0) en = 1;
                        else if (en) break;
                printf("\n");
        num_merge = 0;
        arc_merge();
        if ( num_merge >0 ) {
                io_primitive(1);
                if ( *io_key == 'y' ) {
                        fprintf(io,"\n After merging Arcs:");
                        printf("\nSave data. waiting...");
                        io_primitive(2);
                }
        }

blob_features:
        printf("\n Display Features, click to continue ");
                for (en = 0;;)
                        if (Gcursor(2))
                                if (Gcursor(3,0) == 0) en = 1;
                                else if (en) break;
                printf("\n");
        num_merge = 0;
blob_f_ex:
        blob_feature_extract( bnum );
        io_blob_feature( 1, bnum );

        if ( *io_key == 'y' ) {
                printf("\nSave data. waiting...");
                io_blob_feature(2, bnum );
        }
/* inference */
b_inference:
        blob_inference( bnum );
        if ( flg_retry==1 && Feed_back == 2 && num_merge<Ncp ) {
                if ( blob[bnum].arc_num==2 ) {
                        arc_merge();
                        goto dsp_pr;
                }
                re_merge();
                num_merge++;
dsp_pr:
                io_primitive(1);
                if ( *io_key == 'y' ) {
                        fprintf(io,"\n After re_merging based on inference feed back
information :");
                        printf("\nSave data. waiting...");
                        io_primitive(2);
                }
                printf("\n Feed_back_flag=%d Dif_num_of_line=%d",Feed_back,Dt_lnum);
                goto blob_f_ex;
        }
        printf("\n Another blob Analysis, click to continue ");
```

```
                for (en = 0;;)
                        if (Gcursor(2))
                                if (Gcursor(3,0) == 0) en = 1;

                                else if (en) break;
                printf("\n");

        }
        if ( num_st==3 )
        goto display;
        if ( Nwhite> 1 )
        {
                num_st = 3;
                Num = Nwhite;
                goto blob_check;
        }
display:
        /* display image */

                Vdispl(&fbu[0],0,0,0);
                showbut(b,BUBCOL,BUTCOL);
                Gcursor(4,&svpos);
                Gcursor(1);
                break;
        }
        return 0;
}



but_t buttons[] = {
{       "Quit",         b_quit,   { 10, 10, 25, 100 }, },
{       "Live",         b_live,   { 40, 10, 25, 100 }, },
{       "Histogram",    b_histogram,    { 70, 10, 25, 100 }, },
{       "Blob Chk",     b_blob_analysis,{ 100, 10, 25, 100 }, },
{       "Object Chk",   obj_analysis,   { 130, 10, 25, 100 }, },
{       "Disp Gen_Base",        gen_display,    { 160, 10, 25, 100 }, },
{       "Disp Obj_Base",obj_display,    { 190, 10, 25, 100 }, },
{       "Comp Edge",    b_edge_compare, { 220,10, 25, 100 },},
{       "Add or Load Gen_base",add_blob,{ 250,10, 25, 120 },},
};
#define  NBUTTONS       (sizeof(buttons)/sizeof(buttons[0]))

main()
{
        register but_t      *b;
        register char       *p;
        register  n, key, en;
        coor_t              pos;
        win_t               w[20];
        char                buffer[20];

        IAPDSPD();
        IAPCOLE();
        setupluts();
```

```
        WI(&fbu[0], 0, 0,512,512);      /*      display */
        WI(&fbu[1], 0,512,512,512);     /*      work 1 */
        WI(&fbu[2],512, 0,512,512);     /*      work 2 */
        WI(&fbu[3],512,512, 0, 0);      /*      camera */
        Vfill(&fbu[0],BGCOL);
        Vfill(&fbu[3],0);
        Vcam(0);
        Vxsnap(2,&fbu[3]);
        WI(&w[0],0,256,fbu[3].w_nrow/2,fbu[3].w_ncol/2);
        WM(&w[0],5,0);
        Vreduc(&fbu[3],LUT_F2D,&w[0],2);
/* Input file name */
        printf("\n Save Result in a file?(y/n): ");
        read(0,io_key,1);
        if ( *io_key == 'n')
                goto run_command;

        do {
                printf("\nInput filename: ");
                getstr(buffer,20);
        } while ( buffer[0] == '\0' );

        filename = buffer;
        printf("\n File name:%s\n",filename);

        if ( (io=fopen(filename,"w")) == NULL )
        {
                printf(" File is not open.");
        }
        fprintf(io,"\n File name :%s",filename);
/***run commands ***/
run_command:
        Vdispl(&fbu[0],0,0,0);
        for (Gcursor(en = -1), term = 0; !term; ) {
                if (en < 0) {
                        for (en = 0, b = buttons, n = NBUTTONS; --n != -1; b++)
                                (*b->b_proc)(F_RFSH,b);
                }
                else if (Gcursor(2)) {
                        if (!(key = Gcursor(3,&pos))) en = 1;
                        else if (en && key & 0x07)
                                for (en = 0, b = buttons, n = NBUTTONS; --n != -1; b++) {
                                        if (pinside(&pos,&b->b_win)) {
                                                (*b->b_proc)(F_EXEC,b,fbu,w);
                                                break;
                                        }
                                }
                }
        }
        Gcursor(0);
        if ( *io_key == 'y')
                fclose(io);

}
```

APPENDIX A

File Name: dbase.c

```
#include <v.h>
#include <macro.h>
#include        <stdio.h>

#define         Of3     512
#define         CROW    256+Of3
#define         CCOL    256+Of3

/*extern showbut( but_t *b, int bcol, int tcol );*/
#define         COR             0
#define         X               -1
extern          int             flg_triangle,flg_quart,flg_retry;
extern          int             Nblack,Nobj;
extern          Gen_table       blob[256];
extern          Obj_table       object[10];
extern          short           btree[256][4];

extern          FILE    *io;
extern          char    *io_key;

Gen_table gen_base[] = {
{ 1, "CIRCLE",  1, 0, 0, 0, 0, 0, 0, 0, 4*PI, 1,    PI*120*120, 240*PI, 256+Of3, 256+Of3 },
{ 2, "SQUARE",  0, 4, 0, 4, 4, 4, 0, 0, 16, 0.7854,256*256,    4*256,  256+Of3, 256+Of3 },
{ 3, "RECTANGLE",0,4, 0, 2, 4, 4, 0, 0, X, X,256*160,    832,            256+Of3, 256+Of3 },
{ 4, "SIX_GON", 0, 6, 0, 6, 6, 0, 6, 0, 13.86, 0.9066,25980,     600,    256+Of3, 256+Of3 },
{ 5, "ACUTE_TRIANGLE",0,3, 0, 0, 0, 0, 0, 0, 3, X, X, X, X, X, X},
{ 6, "OBLIQUE_TRIANGLE",0,3, 0, 0, 0, 0, 0, 1, 2, X, X, X, X, X, X },
{ 7, "ISO_TRIANGLE",0,3, 0, 2, 2, 0, X, X, X, X, X, X, X, X },
{ 8, "EQ_TRIANGLE",0,3, 0, 3, 3, 0, 0, 3, 20.78, 0.605, X, X, X, X },
{ 9, "RT_TRIANGLE",0,3, 0, X, X, 1, 0, 2, X, X, X, X, X, X },
{10, "WINDOW",  0,3, 1, X, 2, 2, 0, 0, X, X, X, X, X, X },
{11, "PARRELL",  0,4, 0, 2, 2, 0, 2, 2, X, X, X, X, X, X },
{12, "LADDER",  0,4, 0, 2, 2, 2, 1, 1, X, X, X, X, X, X },
};

Obj_table obj_base[] = {
{ 1, "SQUARE_CIRCLE", 1, {2, 1,X,X}, {5.09,X,X,X} },
{ 2, "SQUARE_SQUARE", 1, {2, 2,X,X},{ 4,X,X,X } },
{ 3, "TRI_TRI",       1, {8, 8,X,X},{ 9.47,X,X,X} },
{ 4, "CIRCLE_CIRCLE", 1, {1, 1,X,X},{ 4,X,X,X} },
{ 5, "SIX_CIRCLE",    1, {4, 1,X,X},{ 3.31,X,X,X} },
};


#define  NGEN_BASE    (sizeof(gen_base)/sizeof(gen_base[0]))
#define  NOBJ_BASE    (sizeof(obj_base)/sizeof(obj_base[0]))

coor_t   tri_1[4] =
{
        { 128+Of3,256+Of3 },
```

```
        { 400+Of3,128+Of3 },
        { 256+Of3,400+Of3 },
        { 128+Of3,256+Of3 },
};

coor_t tri_2[4] =
{
        { 128+Of3,256+Of3 },
        { 400+Of3,100+Of3 },
        { 120+Of3,400+Of3 },
        { 128+Of3,256+Of3 },
};

coor_t tri_3[4] =
{
        { 128+Of3,256+Of3 },
        { 400+Of3,156+Of3 },
        { 400+Of3,356+Of3 },
        { 128+Of3,256+Of3 },
};

coor_t tri_4[4] =
{
        { 128+Of3,256+Of3 },
        { 302+Of3,156+Of3 },
        { 302+Of3,356+Of3 },
        { 128+Of3,256+Of3 },
};

/* triangle with side 200 */
coor_t tri_41[4] =
{
        { CROW-116, CCOL },
        { CROW+58, CCOL-100 },
        { CROW+58, CCOL+100 },
        { CROW-116, CCOL },
};

/* triangle with side 100 */
coor_t tri_42[4] =
{
        { CROW-58, CCOL },
        { CROW+28, CCOL-50 },
        { CROW+28, CCOL+50 },
        { CROW-58, CCOL },
};

coor_t tri_5[4] =
{
        { 128+Of3,200+Of3 },
        { 400+Of3,200+Of3 },
        { 400+Of3,300+Of3 },
        { 128+Of3,200+Of3 },
};
coor_t six[7] =
```

```
{
        { 256+Of3, 156+Of3 },
        { 170+Of3, 206+Of3 },
        { 170+Of3, 306+Of3 },
        { 256+Of3, 356+Of3 },
        { 344+Of3, 306+Of3 },
        { 344+Of3, 206+Of3 },
        { 256+Of3, 156+Of3 },
};

coor_t parrel[5] =
{
        { 176+Of3, 220+Of3 },
        { 176+Of3, 476+Of3 },
        { 336+Of3, 384+Of3 },
        { 336+Of3, 128+Of3 },
        { 176+Of3, 220+Of3 },
};

coor_t ladder[5] =
{
        { 176+Of3, 176+Of3 },
        { 176+Of3, 336+Of3 },
        { 336+Of3, 436+Of3 },
        { 336+Of3, 176+Of3 },
        { 176+Of3, 176+Of3 },
};

coor_t square1[5] =
{
        { CROW-128, CCOL-128 },
        { CROW-128, CCOL+128 },
        { CROW+128, CCOL+128 },
        { CROW+128, CCOL-128 },
        { CROW-128, CCOL-128 },
};

coor_t square2[5] =
{
        { CROW-64, CCOL-64 },
        { CROW-64, CCOL+64 },
        { CROW+64, CCOL+64 },
        { CROW+64, CCOL-64 },
        { CROW-64, CCOL-64 },
};

coor_t six1[7] =
{
        { CROW,    CCOL-128 },
        { CROW-110, CCOL-64 },
        { CROW-110, CCOL+64 },
        { CROW,    CCOL+128 },
        { CROW+110, CCOL+64 },
        { CROW+110, CCOL-64 },
        { CROW,    CCOL-128 },
```

```
};

draw_circle( int Crow,int Ccol, int r)
{
        Gcircle(1, Crow+Of3, Ccol+Of3,r,COR,1);
        return;
}

draw_square( int Orow,int Ocol, int side)
{
        win_t    Square;
        Vwin( &Square, Orow+Of3, Ocol+Of3,side,side);
        G_rect(1,&Square,COR,1);
        return;
}

draw_rectangle( int Orow, int Ocol, int Rside,int Cside)
{
        win_t    Rectangle;
        Vwin( &Rectangle, Orow+Of3, Ocol+Of3,Rside,Cside);
        G_rect(1,&Rectangle,COR,1);
        return;

}

draw_six()
{
        coor_t   *pt;
        pt = &six[0];
        Gpoly(1,pt,7,COR,1);
        return;
}

draw_parrel()
{
        coor_t   *pt;
        pt = &parrel[0];
        Gpoly(1,pt,5,COR,1);
        return;
}

draw_ladder()
{
        coor_t   *pt;
        pt = &ladder[0];
        Gpoly(1,pt,5,COR,1);
        return;
}
draw_window()
{
        coor_t   *pt;
        win_t    iner;

        Vwin( &iner, 256+Of3, 194+Of3,126,124);
```

```
        draw_circle( 256,256, 64 );
        draw_square( 256,192,128 );
        Vfill(&iner,255);
        return;
}
draw_triangle( int n )
{
        coor_t  *pt;
        switch(n)
        {
                case 1: pt = &tri_1[0]; break;
                case 2: pt = &tri_2[0]; break;
                case 3: pt = &tri_3[0]; break;
                case 4: pt = &tri_4[0];

        G_line(1,tri_4[1].c_row,tri_4[1].c_col,tri_4[2].c_row,tri_4[2].c_col,COR);
                        break;
                case 5: pt = &tri_5[0]; break;
        }
        Gpoly(1,pt,4,COR,1);
        return;
}

draw_obj1()
{
        coor_t *pt;
        pt = &square1[0];
        Gpoly(1,pt,5,COR,1);
        Gcircle(1,CROW,CCOL,64,COR,1);
        return;
}

draw_obj2()
{
        coor_t *pt;
        pt = &square1[0];
        Gpoly(1,pt,5,COR,1);
        pt = &square2[0];
        Gpoly(1,pt,5,COR,1);
        return;
}

draw_obj3()
{
        coor_t *pt;
        pt = &tri_41[0];
        Gpoly(1,pt,4,COR,1);
        pt = &tri_42[0];
        Gpoly(1,pt,4,COR,1);
        return;
}

draw_obj4()
{
        Gcircle(1,CROW,CCOL,128,COR,1);
```

```
        Gcircle(1,CROW,CCOL,64,COR,1);
        return;
}

draw_obj5()
{
        draw_six();
        Gcircle(1,CROW,CCOL,64,COR,1);
        return;
}

display_gtable( int m, int n )
{
        Gen_table       *ptr;
        Obj_table       *pt;
        switch( m )
        {
                case 1:  ptr = &gen_base[n];
                         printf("\n Generic Shap No:%d %s ", ptr->no, ptr->name);
                         break;
                case 2:  ptr = &blob[n];
                         printf("\n Blob[%d] Supposed name: %s ", n, ptr->name);
                         break;
                case 3: ptr = &blob[n];
                         printf("\n Supposed Object name:%s", ptr->name);
                         return 1;

        }
        printf("\n Line Number: %d      \t Arc Number: %d ",ptr->line_num,ptr->arc_num);
        printf("\n Equal line number: %d \t Equal Angle Number: %d",ptr->eql_num,ptr-
>eqa_num);
        printf("\n Number of angle with 90: %d \t Big than 90: %d \t Less than 90: %d",ptr-
>eq_90,ptr->gt_90,ptr->les_90);
        printf("\n Compact Ratio= %f \t Thinness Ratio= %f",ptr->compct,ptr->thin);
        printf("\n Area= %d Perimeter= %d \t Gravity Center:(%d, %d)",ptr->area, ptr->peri,ptr-
>xo,ptr->yo);
        printf("\n");
        return 1;
}

save_gtable( int m, int n )
{

        Gen_table       *ptr;
        Obj_table       *pt;
        switch( m )
        {
                case 1:  ptr = &gen_base[n];
                         fprintf(io,"\n\n Generic Shape Base:");
                         fprintf(io,"\n Generic Shap No:%d %s ", ptr->no, ptr->name);
                         break;
                case 2:  ptr = &blob[n];
                         fprintf(io,"\n Blob Analysis Results:");
                         fprintf(io,"\n Blob[%d] Supposed name: %s ", n, ptr->name);
                         break;
```

```
                case 3: ptr = &blob[n];
                        fprintf(io,"\n Object Analysis Results:");
                        fprintf(io,"\n Supposed Object name:%s", ptr->name);
                        return 1;

        }
        fprintf(io,"\n Line Number: %d      \t Arc Number: %d ",ptr->line_num,ptr->arc_num);
        fprintf(io,"\n Equal line number: %d \t Equal Angle Number: %d",ptr->eql_num,ptr-
>eqa_num);
        fprintf(io,"\n Number of angle with 90: %d \t Big than 90: %d \t Less than 90: %d",ptr-
>eq_90,ptr->gt_90,ptr->les_90);
        fprintf(io,"\n Compact Ratio= %f \t Thinness Ratio= %f",ptr->compct,ptr->thin);
        fprintf(io,"\n Area= %d Perimeter= %d \t Gravity Center:(%d, %d)",ptr->area, ptr-
>peri,ptr->xo,ptr->yo);
        fprintf(io,"\n");
        return 1;
}

display_otable( int m, int n)
{
        int             i,k;
        Obj_table       *ptr;

        switch( m )
        {
                case 1: ptr = &obj_base[n];
                        printf("\n Object Shap No:%d %s ", ptr->no, ptr->name);
                        break;
                case 2: ptr = &object[n];
                        printf("\n Object[%d] Supposed name: %s ", n, ptr->name);
                        break;
        }
        k = ptr->gen_no[0];
        printf("\n Object outline shape is %s", gen_base[k-1].name);
        printf("\n It consists of %d Hole(s):",ptr->nhole);
        for ( i=0; i<ptr->nhole; i++) {
                printf("\n\t Area Ratio of Hole %d = %f",i+1,ptr->r_area[i]);
                k = ptr->gen_no[i+1];
                printf("\n\t The hole's generic no = %d is %s ",k,gen_base[k-1].name);
        }
        return;
}


save_otable( int m, int n)
{
        int             i,k;
        Obj_table       *ptr;

        switch( m )
        {
                case 1: ptr = &obj_base[n];
                        fprintf(io,"\n\n Obejct Base:");
                        fprintf(io,"\n Object Shap No:%d %s ", ptr->no, ptr->name);
                        break;
```

```
                case 2:  ptr = &object[n];
                         fprintf(io,"\n Obejct Analysis Results:");
                         fprintf(io,"\n Object[%d] Supposed name: %s ", n, ptr->name);
                         break;
        }
        k = ptr->gen_no[0];
        fprintf(io,"\n Object outline shape is %s", gen_base[k-1].name);
        fprintf(io,"\n It consists of %d Hole(s):",ptr->nhole);
        for ( i=0; i<ptr->nhole; i++) {
                fprintf(io,"\n\t Area Ratio of Hole %d = %f" i+1,ptr->r_area[i]);
                k = ptr->gen_no[i+1];
                fprintf(io,"\n\t The hole's generic no = %d is %s ",k,gen_base[k-1].name);
        }
        return;
}

ge~_display(flg,b,fbu,w)
        but_t           *b;
        win_t   *fbu,*w;
{
        Gen_table       *gen_ptr;
        register  en, key;
        coor_t          svpos;
        int     i;
        char    buffer[20], *filename, *head,*tail;

        switch (flg) {
        case F_RFSH: showbut(b,BUBCOL,BUTCOL); break;
        case F_EXEC:
                Gcursor(0);
                Gcursor(3,&svpos);
                showbut(b,BUHCOL,BUTCOL);

                        printf("  Click to continue ");
                        for (en = 0;;)
                                if (Gcursor(2))
                                        if (Gcursor(3,0) == 0) en = 1;
                                        else if (en) break;
                        printf("\n");

                for ( i=1; i< NGEN_BASE+1; i++ )
                {
                        Vfill(&fbu[3],255);
                        switch(i){
                        case 1: draw_circle( 256,256,120);break;
                        case 2: draw_square( 128,128,256); break;
                        case 3: draw_rectangle( 128,128,160,256); break;
                        case 4: draw_six(); break;
                        case 5: draw_triangle(1); break;
                        case 6: draw_triangle(2); break;
                        case 7: draw_triangle(3); break;
                        case 8: draw_triangle(4); break;
                        case 9: draw_triangle(5); break;
                        case 10:draw_window();
                                break;
```

```
                        case 11:draw_parrel();      break;
                        case 12: draw_ladder(); break;
                        }

                        Vreduc(&fbu[3],LUT_F2D,&w[0],2);
                        Vdispl(&w[0],0,w[0].w_row,w[0].w_col);
                        display_gtable(1,i-1);
                        printf(" Click to continue ");
                        for (en = 0;;)
                                if (Gcursor(2))
                                        if (Gcursor(3,0) == 0) en = 1;
                                        else if (en) break;
                        printf("\n");
                }
                showbut(b,BUBCOL,BUTCOL);
                Gcursor(4,&svpos);
                Gcursor(1);
                break;
        }
        return;
}

obj_display(flg,b,fbu,w)
        but_t           *b;
        win_t   *fbu,*w;
{
        Obj_table       *obj_ptr;
        register  en, key;
        coor_t          svpos;
        int     i;

        switch (flg) {
        case F_RFSH: showbut(b,BUBCOL,BUTCOL); break;
        case F_EXEC:
                Gcursor(0);
                Gcursor(3,&svpos);
                showbut(b,BUHCOL,BUTCOL);

                for ( i=1; i< NOBJ_BASE+1; i++ )
                {
                        Vfill(&fbu[3],255);
                        switch(i){
                        case 1: draw_obj1();break;
                        case 2: draw_obj2(); break;
                        case 3: draw_obj3(); break;
                        case 4: draw_obj4(); break;
                        case 5: draw_obj5(); break;
                        }

                        Vreduc(&fbu[3],LUT_F2D,&w[0],2);
                        Vdispl(&w[0],0,w[0].w_row,w[0].w_col);
                        display_otable(1,i-1);

                        printf(" Click to continue ");
                        for (en = 0;;)
```

```
                                if (Gcursor(2))
                                        if (Gcursor(3,0) == 0) en = 1;
                                        else if (en) break;
                        printf("\n");
                }
                showbut(b,BUBCOL,BUTCOL);
                Gcursor(4,&svpos);
                Gcursor(1);
                break;
        }
        return;
}

compute_obj_feature( int nhole, int m, int i_blob,int i_obj)
{
        Obj_table       *pt;
        float           rarea;

        pt = &object[i_obj];
        rarea =((float)( blob[i_blob].area))/((float)(blob[m].area));
        pt->r_area[nhole] = rarea;
        return;
}

obj_feature_extract()
{
        int     i,j,m,nhole;

        Nobj = Nblack-1;
        i = 2;
        for ( j=1; j<Nobj+1; j++ )
        {
                nhole = 0;
                object[j].gen_no[0] = blob[i].no;
                m = btree[i][2];
                if ( m == -1 )
                {
                        object[j].nhole=0;
                        object[j].name = blob[i].name;
                        continue;
                }else {
right_sibling:
                        nhole++;
                        object[j].gen_no[nhole] = blob[m].no;
                        compute_obj_feature( nhole,m,i,j );
                        m = btree[m][0];
                        if ( m == -1 )
                        {
                                object[j].nhole = nhole;
                                continue;
                        } else
                                goto right_sibling;
                }
        i+=2;
        }
```