An Architecture for the Recognition of Rigid Body Translation

By Wong Hin Lau

A thesis submitted to

The Department of Computer Science
The Chinese University of Hong Kong

in partial fulfillment of the
requirements for the degree of

Master of Philosophy

Date : August, 1993

UL

thesis
TA
1637
W66
1993

Abstract

In this thesis, an architecture for the detection of rigid body translation has been proposed. The methods for the detection of motion in the horizontal (x), vertical (y) and the combination of both directions have been presented. In the proposed model, there are 4 processing layers known as Bit Map Image Recording Layer (Layer 1), Edge Recording Layer (Layer 2), Attribute Recording Layer (Layer 3) and Motion Detecting and Recording Layer (Layer 4) respectively.

Between layer 1 and layer 2, edge detection is performed using the mask(s) described in chapter 3. Between layer 2 and layer 3, the edge features obtained and recorded in layer 2 are classified by an edge feature classification matrix into an integer as the feature number, this feature number is unique for a specified pattern but is able for inexact matching. Feature classification and feature matching will be described in chapter 4. Between layer 3 and layer 4, feature matching is carried out to produce the primary matching results by using the operator, EXCLUSIVE OR, and this is also described in chapter 4. The building of object motion correspondences involves the segmentation of the input image, a new technique called edge directed image segmentation is described in chapter 5. The treatment of mismatch and the techniques for building object correspondences by using subset illumination and image segmentation are discussed in chapter 6. Finally the results obtained from real pictures, the analysis of the results and the discussion are given in chapter 7.

CHAPTER 1

INTRODUCTION TO DIGITAL IMAGE PROCESSING and RIGID BODY TRANSLATION DETECTION

1.1. Digital Image Processing and Applications

Digital image processing usually refers to the processing of a two dimensional digital image by a digital computer. A digital image is usually an array of real numbers (most likely integers) represented by a finite number of bits. An image given in the form of a slide, photograph, transparency or chart is first digitized and stored as an array of binary digits in computer memory. The digitized image is usually referred as **digital image**. This type of images can be displayed by a high resolution monitor with some programs for the transformations in order to extract some information or to make it a different view. The processing control by a digital computer of digitization, storage, transformation and display operations is usually called **Digital Image Processing**.

Digital image processing has a broad spectrum of applications, such as remote sensing via satellites and other spacecraft, image transmission and storage for business applications, medical processing, radar, sonar, and acoustic image processing, robotics, automatic navigation and automatic inspection of industrial parts, etc..[Ref. 1.1]

Images acquired by satellites are useful in tracking of earth resources; geographical mapping, prediction of agricultural crops, urban growth, weather; flood and fore control, and many other environmental applications. Space applications include recognition and analysis of objects contained in images. Image transmission and storage applications occur in broadcast television, teleconferencing, transmission of facsimile images for office automation, communication over computer networks etc.. Another important application of image processing is for **Dynamic Scene Analysis** that involves feature extraction, feature classification, feature matching, object recognition, image understanding and knowledge extraction from a series of digital images. The sucessful results of dynamic scene analysis can be used to build robots and automatic machines, etc..

Although there are many image applications and problems, in this thesis, only one of the topics for dynamic scene analysis, or more accurately, the recognition of rigid body translation will be considered. And the formulation of this topic assumes that a series of consecutive images are already obtained and digitized with an object in the image that have performed translation motion. The final goal is to find out the quantity of motion in horizontal (x-direction) and vertical (y-direction) or the combination of the both directions between 2 consecutive images.

1.2 The Problems and Assumptions

The main problems for the recognition of rigid body translation can be simply described as that given a series of images taken at various consecutive time intervals, there are some objects in the given images (known as frames) that have performed translation(s), the motion in horizontal direction (x-direction), in vertical direction (y-direction) or the combination of the both. Moreover, since the objects being considered are rigid bodies, the shapes and the intensities are assumed to be unchanged after performed the translation action. The problem is to find out the motion quantities for the objects in the images.

The main assumptions in these problems are that the objects being considered are rigid bodies, the motion activities are translations, motion in horizontal direction (x-direction), in vertical direction (y-direction) or the combination of the both. Since the objects are of rigid bodies, we can further assume that the intensities, the shapes, the shadows and other geometric structures are unchanged after the translation.

This problem, seems to be simple, but there is no acceptable solution so far. It has drawn many researchers recently. In next two sections(Section 1.3 and Section 1.4), some current approaches will be described and commented.

1.3 The Research Background

The techniques for the recognition of rigid body translation involves edge detection, feature classification, feature matching, noise minimization, image segmentation and motion quantitization. First, the techniques of edge detection are well developed and will be discussed in chapter 3.

For feature classification, although there are many existing techniques, they are not very suitable for the application of low level processing to generate high level results. The existing techniques can be classified into two main categories, statistical and structural [Ref. 1.2].

For the statistical approach, the existing techniques include Image Transform Methods, Edge Density Methods, Histogram Features, etc..[Ref. 1.3]

In the structural approach[Ref. 1.4] [Ref. 1.5], the purely structural textures are assumed as deterministic texels, which repeat according to some placement rules, either deterministic or random. A texel is isolated by identifying a group of pixels having certain invariant properties, which repeats in the given image. The texel may be identified by its gray level, shape, or homogeneity of some local property, such as size, orientation, or histogram. The placement rules define the spatial relationships between the texels. These spatial relationships may be expressed in terms of adjacency, closest distance, periodicities and so on, in the deterministic placement rules. In such cases, the texture is labeled as being strong. For randomly placed texels, the associated texture is called weak and the placement rules may be expressed in terms of measures such as the following :

1. Edge Density
2. Run lengths of maximally connected texels
3. Relative extrema density, which is the number of pixels per unit area showing gray levels that are locally maxima or minima relative to their neighbors.

For feature matching, it is the most important part in dynamic scene analysis. There are generally two approaches. One is to transform a number of related pixels into a single high level feature, matching is then carried out in the transformed high level feature. This approach is known as high level feature matching. Another approach is that matching is carried out in the low level, some techniques are then applied to integrate the low level matching results for generating an overall high level result.

A few existing techniques are including :

1.3.1 Image Subtraction

Changes in a dynamic scene observed as $u_i(m,n)$ ($\mathbf{u_i(m,n)}$ is a representation of image of size **mxn**), i=1,2,3,.... are given by

$$e_i(m,n) = u_i(m,n) - u_{i-1}(m,n) \quad \ldots\ldots \text{ (EQ 1.1)}$$

This method is elementary but very useful in motion detection, monitoring system, segmentation of parts in a complex assembly. However, it is only an approximate method which is not accurate enough for our application.

1.3.2 Template Matching and Area Correlation

The presence of a known object in a scene can be detected by searching for the location of match between the object template *u(m,n)* and the scene *v(m,n)*. Template matching can be conducted by searching the displacement of *u(m,n)* where the mismatch energy is minimum, i.e. it is an optimization problem.

For the displacement quantity(p,q), we can define the mismatch energy :

$$\begin{aligned} e_n^2(p,q) &= \sum_m \sum_n [v(m,n) - u(m-p,n-q)]^2 \quad \ldots\ldots \text{ (EQ 1.2)} \\ &= \sum_m \sum_n |v(m,n)|^2 + \sum_m \sum_n |n(m,n)|^2 - 2\sum_m \sum_n v(m,n)u(m-p,n-q) \end{aligned}$$

The basic idea is to search for the maximum for $2\sum_m \sum_n v(m,n)u(m-p,n-q)$.

This is a statistical method and requires considerably heavy computation power. This method may work but it seems impractical because it still needs a lot of preprocessing before the actual computation can proceed.

Other methods such as *Matched Filtering* and *Direct Search Methods* can also be found in [Ref. 1.6].

Rangarajan and Shah [Ref. 1.7] proposed an algorithm to establish the motion correspondence in 1989 by using statistical approach. The algorithm first identifies the points in the frame $\mathbf{I}_k$ which do not have corresponding points in the frame $\mathbf{I}_{k+1}$, the new feature points in frame $\mathbf{I}_k$ corresponding to the missing points are then created. After that the corresponding matrix are filled. The details of the algorithm are listed in Appendix D.

1.4 Current Literature on the related fields

There are certain literal issues which have been published currently and are highly related to this project. The following subsections are the brief descriptions for them respectively. The comparison of the listed approaches against the approaches in this project will be given in chapter 7.

1.4.1 Grouping Edgels into Structural Entities

James N. Huddleston and Jezekiel Ben-Arie have proposed a method to group edgels into structural entities by using circular symmetry, Hough transform and probabilistic non-accidentalness conditions [Ref. 1.8]. The objective of their research is to group image edge elements (edgels) into structurally significant entities. The basic idea is based on the fact that any continuous (2D) curve can be approximated by a set of 2D segments with constant curvature. In this literature, they choose circular arcs and linear segmentations for the representation of pictorial structural entities.

A new grouping algorithm called the distributed Hough transform has been developed to group edgels into circular features and generating explicit quantitative descriptions of the feature. The principle of proximity weighted circular symmetry has also been incorporated and is based upon non-accidentalness, viewpoint invariance, and recent probabilistic models of projected angles and distances. Since the probabilistic model of projected features [Ref. 1.9 , Ref. 1.10] has been incorporated, it groups edgels into features that are more likely to correspond to objects in the 3-dimensional world.

This literature is related to this project as it can act as the preprocessing of the scenery for the extraction and the representation of the edge features by curvatures. The object correspondences can than be built by comparing the curvature representation of the consecutive scenes.

Their method has some advantages over the conventional Hough transform [Ref. 1.11]. One advantage is that it requires a parameter space of only one dimension for the detection of circular arcs where the conventional one requires a three dimensional space. Another advantage of this issue is that it is an inherently parallel process so that can be implemented in a parallel machine or an artificial neural network. An outline of such a network is represented in [Ref. 1.12].

The major limitation of the approach is that the introduction of orientational noise causes the algorithm to degrade more rapidly than does the human perception of features. That is, it is sensitive to orientational deviations. This is the main reason for our not choosing this method for the implementation in out proposed model.

Some of the related issues for the same problem can be found in [Ref. 1.11], [Ref. 1.13 , Ref. 1.14].

1.4.2 Extracting Geometric Primitives

Gerhard Roth and Martin D. Levine have published a method for the extraction of geometric primitive [Ref. 1.15]. This is closely related for the building and comparing of the features of an object in two image frames so that the object correspondences can be built.

First, the geometric features are extracted in the two image frames being compared, then the extracted featured are classified. Second, comparison has been made to build the feature correspondences. Third, from the comparison results, the object correspondences can be deduced.

[Ref. 1.15] concentrates on the extracting of geometric features from the field of robust statistics. They showed that extracting a single geometric primitive is equivalent to finding the optimum value of a cost function which has potentially many local minima. Besides providing a unifying way of different extracting algorithm, they also showed that for the efficient extraction of the true global minimum had to be found with as few evaluations of the cost function as possible.

The algorithm can be described as :

```
For K randomly chosen set of points
DO
        1. Find the parameter vector of the primitive through the points.
        2. Compute the residuals of the entire set of points.
        3. Rank the goodness of the primitive using the cost function.
        4. Mark the primitive with the smallest cost.
ENDDO
```

Related literature includes the methods for obtaining geometric data such as described in [Ref. 1.16], or by processing passive sensor data by the methods such as stereo vision [Ref. 1.17], or by simple edge detection and thresholding of intensity images [Ref. 1.18] and etc..

Recently, the problem of extracting geometric primitives has been tackled from the point of view of robust statistical, which is called RS approach [Ref. 1.19 -

Ref. 1.21]. Traditional fitting processes, such as least squares, are not capable for primitive extraction because the assumption is made that all the geometric data belongs to a single primitive [Ref. 1.22 - Ref. 1.23].

The advantages of the methods proposed in [Ref. 1.15] include the capability of extracting a wide variety of different geometric primitives and can use many different types of cost functions. It does not require any ordering of the geometric data . It can also be applied to both sparse and dense, single and multi-view geometric data. And can be formulated by using a parallel machine for faster processing.

The main limitation of [Ref. 1.15] is that each time, only a single primitive can be extracted. And there is no guaranty for the computation time which may cause real time processing impossible and hence is not selected for our proposed model.

1.4.3 Current techniques for Texture Segmentation

Image segmentation is used in our model for the final decision of the motion of an object and for the calculation of matching strength. This is particularly important for the removal of matching noise and hence building the correct motion correspondences. Image segmentation plays an very important role in our model. The following is a review for the current techniques. A new method has been developed and described in chapter 5 in this thesis.

The area of texture segmentation has a tremendous growth in recent years. There has been a great deal of activities in both the refinement of the known approaches and the development of completely new techniques. Although there is much work in this field, the main trend is in the development of feature based approaches [Ref. 23].

For the feature-based methods for image segmentation, they include using operator-based features, statistically-based features, transform domain features and etc..

For operator based feature, the basic idea is to use an operator to convolute with the image to be segmented. Statistics, such as the variance, are computed within a mask about each pixel in the resulting images. A function is used to group the results into a homogenous region. The process continues until the whole image has been segmented. The related literature includes Laws method in [Ref. 1.24], a texture operator derived by Conners, Trivedi, and Harlow in [Ref. 1.25]. A simple operator for fast discrimination between textured and uniform regions has been proposed by Dinstein et, al. in [Ref. 1.26]. Methods similar to that proposed by Laws were introduced by User in [Ref. 1.27], Wang, Hanson, and Riseman in [Ref. 1.28].

For statistically based features, the basic idea is to use the spatial gray level dependence as the criteria for the combination of the spatial pixels to form a homogenous region in an image to be segmented. The process continues until the whole image has been segmented. The related issues include [Ref. 1.29] by Haralick, Shanmugam, and Dinstein who have used the gray-tone spatial-dependence matrix for extracting statistical features. A method using both statistical and structural features has been proposed Abele in [Ref. 1.30]. Features for segmentation are selected from a large set of features associated with texture primitives. The selection for merging is performed by a space-invariant, non-linear operation, the purpose of which is to reduce the distance between primitives in the same class, while increasing the distance between different classes. In [Ref. 1.30], Lowtiz proposed that information extracted from local histograms as features for texture segmentation, etc..

For transform domain feature approach, Xu and Fu in [Ref. 1.31] proposed a method which works as that scenes are initially segmented by use of multiple thresholds to reduce the number of gray levels in the image. Segmentation is then performed using co-occurrence matrices and the split and merge algorithm. In [Ref. 1.32], Jernigan and D'Astous propose entropies, calculated for subimages and over various regions of the subimages power spectrum, as a set of feature.

1.4.4 CAD-based vision object recognition

Object recognition is an important problem in computer vision, especially in motion detection, for the building of object correspondences in the given image frames. Farshid Arman and J. K. Aggarwal proposed a method for the recognition of an object in a given scene by using a three-dimensional model of the object [Ref. 1.33]. The scene may contain several overlapping objects, arbitrarily positioned and oriented. A laser range scanner is used to collect three dimensional data points from the scene. The collected data is segmented into surface patches, and then the segments are used to calculate the various 3D surface properties. On the other hand, the CAD models are designed using commercially available CADKEY (a software package) and access via the industrial standard IGES. The models are analyzed off-line to derive various geometric features, their relationships, and their attributes. A strategy for identifying each model is then automatically generated and stored. The stored strategy is then used in run-time to complete the task for object recognition.

This approach, has its advantages. The first one is that it uses commercially available package CAD to design object models which is not an experimental package, and thus the method has a value for real world application. The second one is that IGES has been used as an interface to the CAD system to infer the geometric information necessary for object recognition. Using these two packages reduces the dependency of the vision system on any particular CAD modeler and increased its applicability. The third one is only important features are selected for recognition, not all features. This may increase the efficiency of the system.

However, there are still a number of limitations for this approach. The first one is the chance for mismatching an similar object is increased. This is due to the fact that only important features are selected for recognition, similar objects have similar "important features", without considering other features, mismatching can occur. The second one is that this system can only recognize the objects which are known in detail before the recognition procedure. The third one is that there is no strategy for inexact matching due to noises or hardware equipment errors.

It is the second and third limitations which make it unusable in this thesis because in our situation, the objects being considered may not be known before the recognition, and we assume that there are noises present in the images being considered in our model.

The related literature includes [Ref. 1.34, Ref. 1.35], which give a complete review of object recognition system. Grimson [Ref. 1.36, Ref. 1.37, Ref. 1.38] uses two sets of simple features for matching : linear edges and circular arcs. To account for possible occlusions, null features are introduced and paired with otherwise unmatched features. Flynn and Jain [Ref. 1.39] order the features extracted from the scene by area and type. A series of unary and binary feature constraints are used to reduced the number of possible matches. Fan, Medioni and Nevatia [Ref. 1.40, Ref. 1.41, Ref. 1.41] represent objects and models using attributed graphs. The matching process starts by reducing the possible models based on the number of nodes, the visible 3D area and the number of planner nodes in each graph. Hansen and Henderson [Ref. 1.43, Ref. 1.44] developed a system for the automatic generation of recognition strategies using several feature properties, such as robustness, completeness, consistency, cost, uniqueness and rarity. This method does not depend on specific features; but selects the given features to be used in matching objects to the model based on the above mentioned properties.

1.4.5 Motion field and Optical Flow [Ref. 1.45]

Motion field is the 2-D vector field which is the perspective projection on the image plan of the 3-D velocity field of a moving scene. ***Optical flow*** which is defined as the estimate of the motion field which can be derived from first order variation of the image brightness pattern. The assumption that the motion field and the optical flow coincide has often been made, the intuitive rationale being that this is true when spatial variations in an image bright pattern E correspond to physical features on the visible 3-D surface [Ref. 1.46 - Ref. 1.48]. Horn [Ref. 1.49] however, has pointed out some examples in which this assumption does not hold. Algorithms which deal with the recovery of the motion field from dense optical flow data have been proposed with, usually, the implicit assumption that these two fields are the same [Ref. 1.50 - Ref. 1.52].

In fact, [Ref. 1.45] has shown that *the optical flow and the motion fields are in general different, unless special conditions are satisfied.* In fact, even the hypothesis of a Lambertian reflectance function of the viewed surfaces is not sufficient by itself to guarantee that the two vectors are the same. In fact, where sharp changes in intensity over time are due to physical events on the moving surface, the estimates of the component of the motion field along the direction of the spatial gradient of the image brightness pattern - estimates which can be obtained by means of first order derivative of the image brightness pattern - are accurate[Ref. 1.45]. But these estimates are unlikely to be useful for methods which rely upon a very precise , local reconstruction of the motion field.

The correct usage of optical flow (3-D structure of the viewed scene) is to obtain the qualitative properties of the motion field. That is, only the ***qualitative properties*** of the motion field can be obtained from the qualitative properties of the optical flow. A thorough analysis of this approach - proposed first by [Ref. 1.53, Ref.

1.54] which suggested that if the motion field and the optical flow are sufficiently similar, they also have the same qualitative properties. Therefore, the qualitative properties of optical flow might be useful in recovering motion information [Ref. 1.55].

The major advantage of using optical flow to deduce the motion field is that it may gain statistically fit results. However, this result may not be correct as suggested by [Ref. 1.45] especially for extracting quantitative information, it is dangerous. And, even the qualitative information can be extracted only when the two fields are sufficiently similar. Therefore it is unsuitable for deducing quantitative information of motion field. [Ref. 1.56] suggested a method for determining three-dimensional motion and structure from optical flow generated by several moving objects. But it has the same problem suggested by [Ref. 1.45 Section II].

Thus the deduction of motion from the information of optical flow may be potentially dangerous (it is vital because the mathematical formulation may be incorrect as pointed out by [Ref. 1.45]) and hence is not used in our system.

1.5 The contents of the thesis

In this thesis, a new model for motion or dynamic scene analysis will be proposed. The working mechanisms for the whole model and the various layers will be described. The formulations with mathematical explanations are also included.

In chapter 2, the new model for the recognition of rigid body translation will be described in general. In chapter 3, the edge detection mechanism and its mathematical formulation will be formulated and defined. Moreover, the application of an edge detection mask in the connection in our model will also be illustrated. In Chapter 4, a completely new technique called edge feature classification will be introduced and formulated. The methods of the application of the edge feature classification techniques have been discussed and merged in our model for producing the primary matching results in layer 4 are also illustrated. Feature matching from the results described and matching enhancement techniques will be also described. In chapter 5, the importance of image segmentation and the related techniques will be formulated. In chapter 6, the building of motion correspondences are described. Finally in Chapter 7, some of the experimental results, conclusion and discussion will be given, the directions for further research will be given based on the experiences of this research.

CHAPTER 2

AN ARCHITECTURE FOR THE RECOGNITION OF RIGID BODY TRANSLATION

2.1 Introduction

Vision is the main source of information for the operation of mobile robots and autonomous vehicles. Even if the machines are equipped with the best automatic navigation system, the accumulation of errors due to the change of positions and the computation adjustment will need periodic corrections. Without the dynamic information supplied by the real situation, there is no data to make corrections. Thus, a camera attached to the robot can be a good source of information for making corrections.

Moreover, a fast, accurate method of image analysis can provide information for the decision making of the robot so that it can avoid the collision with the other moving objects. If the results from image analysis is fast and reliable enough, it is possible to build an autonomous mobile machine. Everything can be determined with one or a pair of stereo cameras installed to the machine plus the image analysis tools attached.

The analysis of a dynamic scene for the recognition of rigid body translation involves the detection of motion quantities in the horizontal (x) direction, vertical (y) direction and the combination of the both directions. An other equally important aspect is the detection of the shape(s) of the objects being detected and considered.

Within this thesis, the architecture for motion detection will be presented. The method for translation detection in both horizontal (x) and vertical (y) direction and the combination of the both will be analyzed.

2.2 The Motion Detection Architecture

The Pixel Based Part of Motion Detection Architecture for the recognition of rigid body translation consists of four processing layers. Each layer consists of an array of processing elements which form an array. The interconnection of the cells of one layer to other layer(s) forms the processing logic. Basically, the functions of the cells in the same layer are the same, the interconnection and the processing criteria are also the same. The 4 layers are named as :

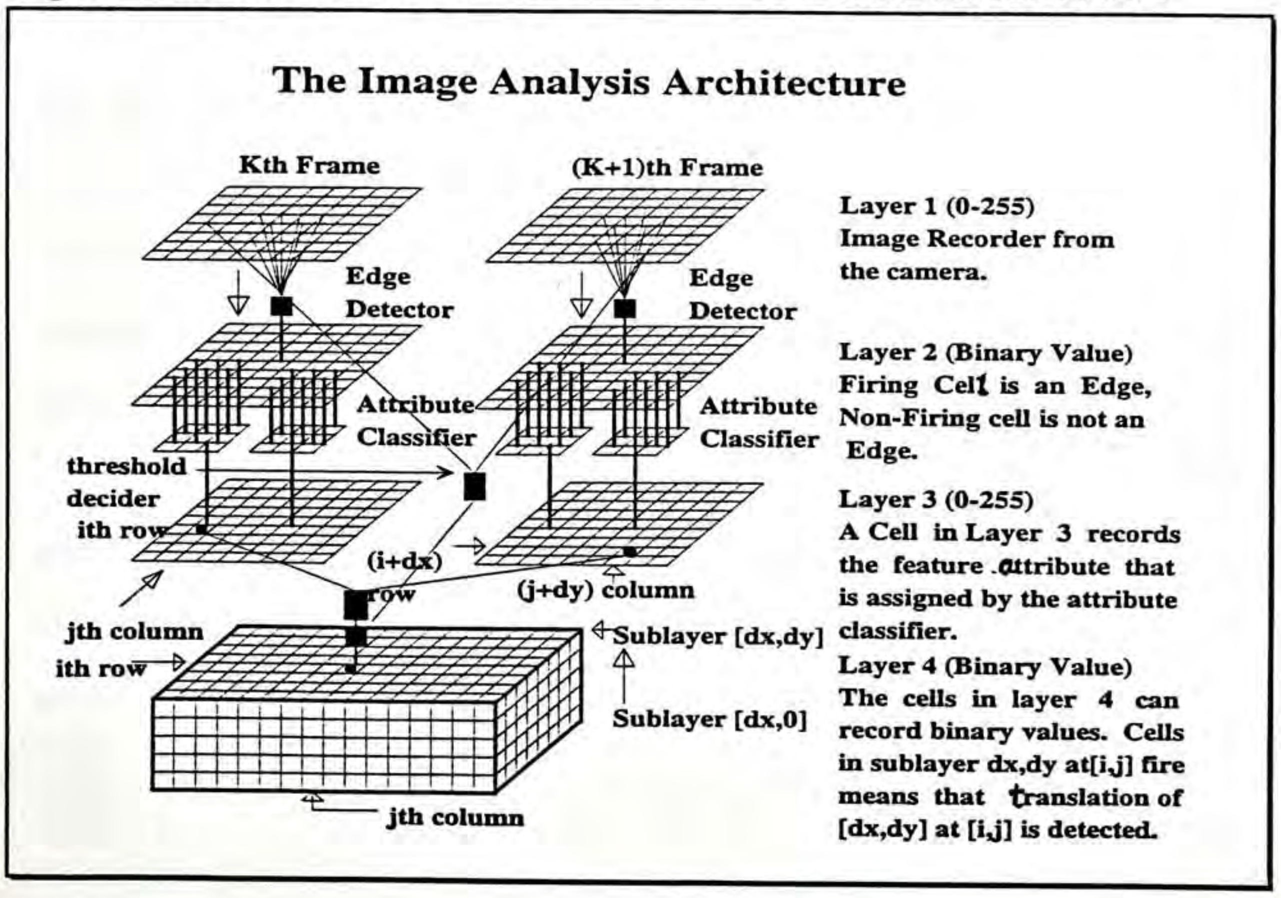
(a) Bit Map Image Recording Layer (Layer 1) ;

(b) Edge Recording Layer (Layer 2) ;

(c) Attribute Recording Layer (Layer 3) ;

(d) Translation Detection and Recording Layer (Layer 4) .

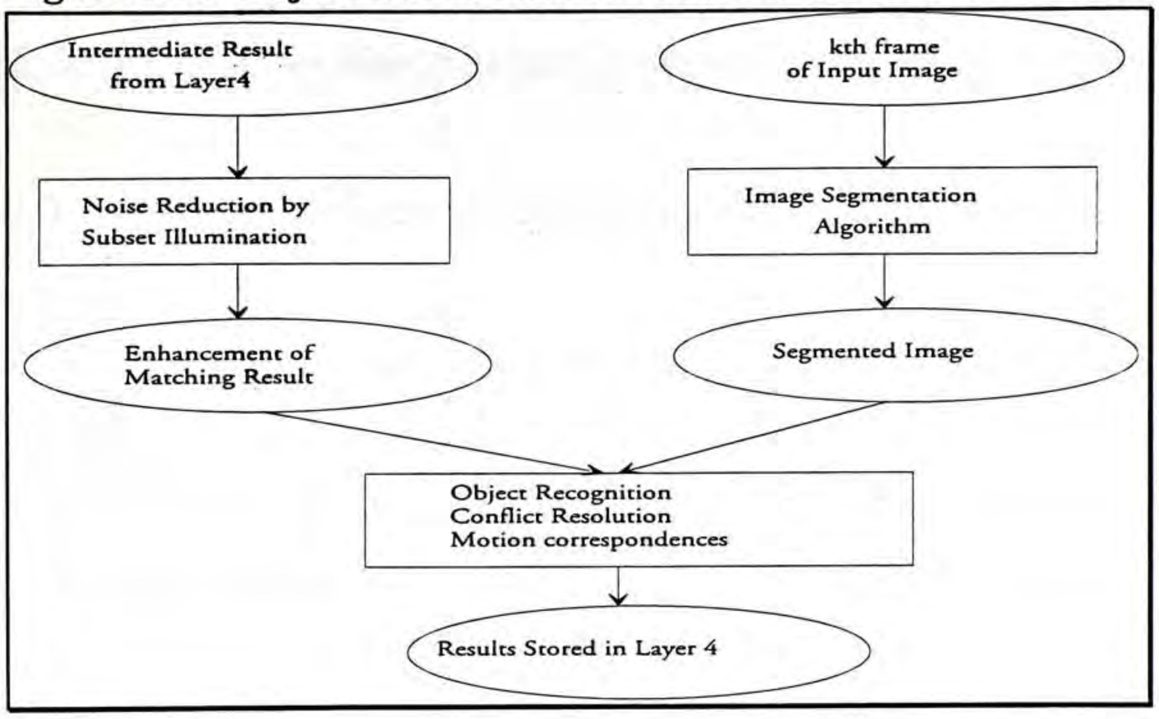
Figure 2.1a Pixel Based Part of Motion Detection Architecture


The Image Analysis Architecture
Kth Frame
(K+1)th Frame
Edge Detector
Edge Detector
Attribute Classifier
Attribute Classifier
threshold decider
ith row
(i+dx) row
(j+dy) column
jth column
ith row
Sublayer [dx,dy]
Sublayer [dx,0]
jth column
Layer 1 (0-255) Image Recorder from the camera.
Layer 2 (Binary Value) Firing Cell is an Edge, Non-Firing cell is not an Edge.
Layer 3 (0-255) A Cell in Layer 3 records the feature attribute that is assigned by the attribute classifier.
Layer 4 (Binary Value) The cells in layer 4 can record binary values. Cells in sublayer dx,dy at[i,j] fire means that translation of [dx,dy] at [i,j] is detected.

As shown by Figure 2.1a, the layered structure of the Pixel Based Motion Detection Architecture is organized in such a way that the input image frames are passed through Layer 1 first, where the gray level can be recorded. Since most of the image analysis techniques need the input of 2 images simultaneously (such as stereo images, time varying dynamic scenes, etc.), the architecture has dual parts of Layer 1 to Layer 3 which are identical.

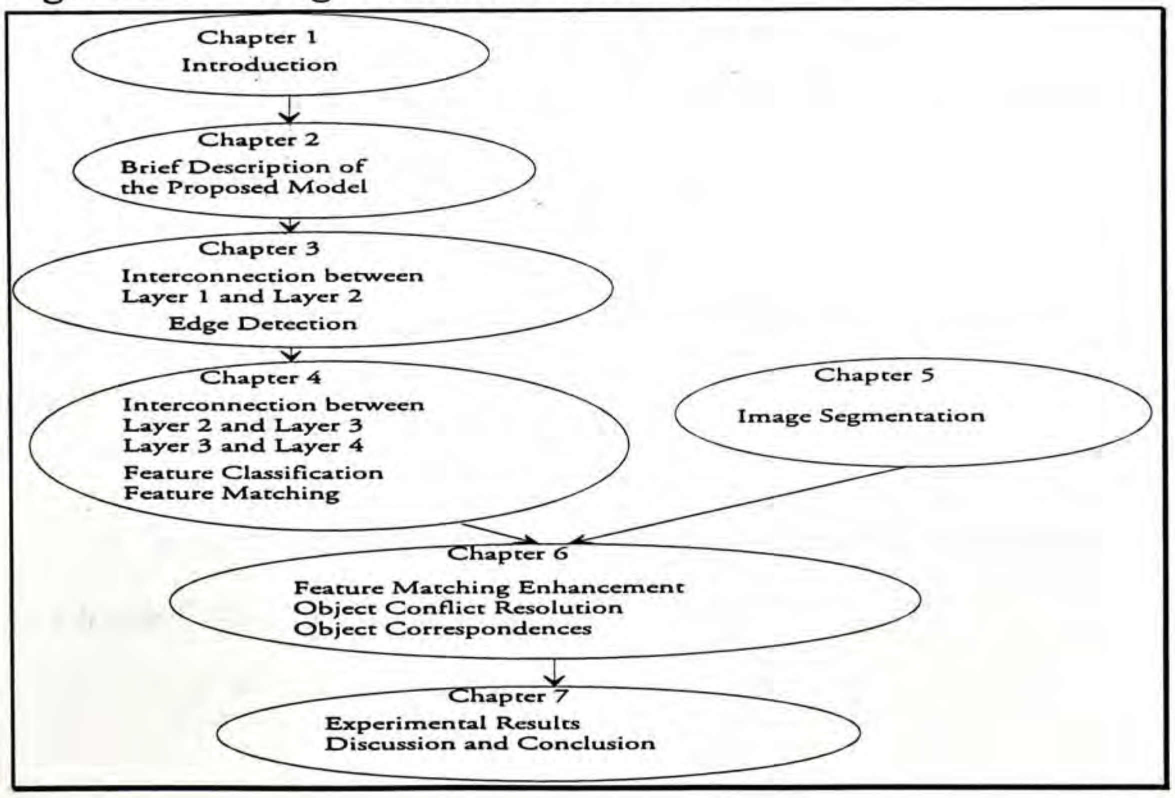
The proposed architecture functions in such a way that the contents of Layer 1 go directly from the camera or frame grabber (in case of stereo images, there are a pair of dual cameras; for time varying dynamic scene analysis, the left component takes the input at time t, while the right component takes the input at time t+dt). Layer 2 takes the input of Layer 1 and performs edge detection. Between Layer 1 and Layer 2, there is an ***edge detector***. Layer 3 takes the input of Layer 2. Between Layer 2 and Layer 3, there is an ***attribute classifier***. Layer 4 takes the input of Layer 3 and Layer 1. In between Layer 3 and Layer4, there is an ***attribute identifier***. In between Layer 1 and layer 4, there is a ***gray level thresholder***. Layer 4 takes the input of Layer 3 as the initial value, it then performs the matching activities in the feature level. Up to now, all detection mechanism has been completed in pixel level by Figure 2.1a. Since there may be a lot of mismatch errors in the feature level, a technique called subset illumination is applied to minimize feature matching error. Image segmentation is carried out at the same time for the image taken at time = $\mathbf{t_n}$. Finally object correspondences can be built from the current processing results of Layer 4 and the input of the segmented image. This type of processing is carried out in object level and is carried out as shown by Figure 2.1b. With the inputs of the raw matching results in Layer4 and the segmented image, the object level processing then visits each homogeneous regions, the results of processing will be reflected on the matching results in layer 4.

Figure 2.1b Object Based Part of Motion Detection Architecture


Intermediate Result from Layer4
kth frame of Input Image
Noise Reduction by Subset Illumination
Image Segmentation Algorithm
Enhancement of Matching Result
Segmented Image
Object Recognition
Conflict Resolution
Motion correspondences
Results Stored in Layer 4

In next section, we will give a brief description for each layer, the interlayer connection and the mathematical formulation for the pixel based part of motion detection architecture. The object based part of motion detection architecture will be discussed in Chapter 5 and Chapter 6 in detail. The organization for the whole thesis is as shown in Figure 2.1c.

Figure 2.1c Organization of this thesis



2.3. Interlayer Mechanism and Mathematical Formulation

2.3.1 Interconnection and Functioning Between Layer 1 and Layer 2

As shown by Figure 2.1a, Layer 1 is actually responsible for the recording of the image bit map. A collection of cells in Layer 1 forms a digital image. Now suppose a cell in Layer 1 is able to record discrete value from 0 to 255 of which must be an integer, then Layer 1 is recording a digital image with gray level ranging from 0 to 255.

Layer 2 records the change of intensity (gray level) of the digital image recorded in Layer1. The relation of Layer 2 with Layer 1 is defined as :

$$\text{Layer2}[i,j] = f_{21}\left(\sum_{m=i-1}^{i+1} \sum_{n=j-1}^{j+1} (a_{m-i+2,n-j+2}) * \text{Layer1}[m,n] \right) \ldots \text{(EQ 2.1)}$$

Where f_{21} is a function that maps to a binary value either 0 or 1;

Layer2[i,j] is a cell in Layer 2 at position [i,j];

Layer1[m,n] is a cell in Layer 1 at position [m,n];

a_{xy} is an element of a 3x3 matrix A used for convolution with

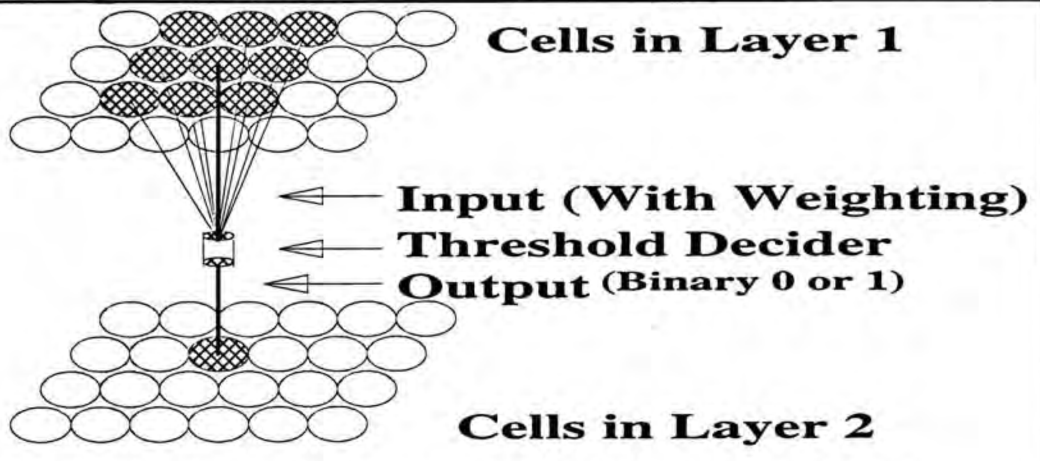
$x \in \{1,2,3\}$ and $y \in \{1,2,3\}$.

We can take the matrix A to be one of the famous Laplacian Operators which are listed in Table 3.3 :

The details of the edge detection mask **A** for digital images will be discussed in Chapter 3 for more detail.

Therefore, the interlayer connection between Layer 1 and Layer 2 can be simply depicted in Figure 2.2.

Figure 2.2 Interlayer Connection Between Layer 1 and Layer 2



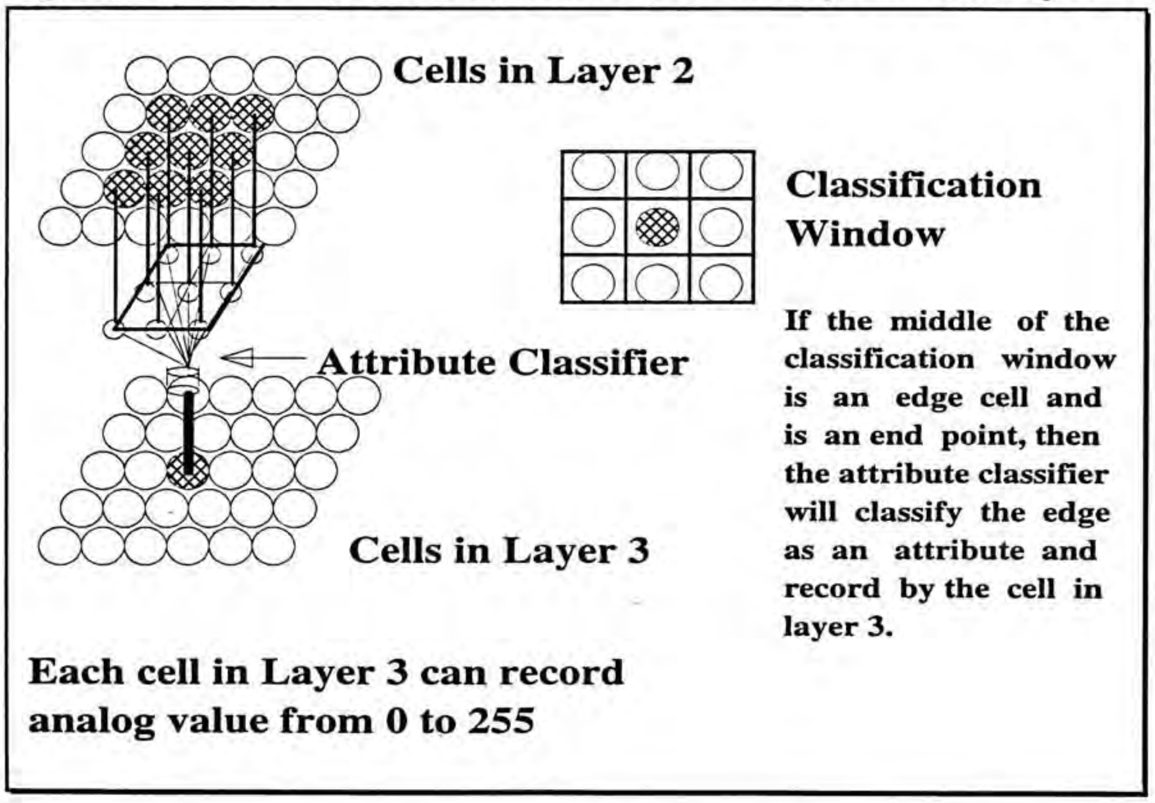
The Relationship between cells in layer 1 and cells in layer2 is the edge detection. Every cell in layer 2 is connected to 9 cells in layer 1 in corresponding positions as shown above.

As shown in Figure 2.2, a cell at position [i,j] in Layer 2 is connected to 9 cells in Layer1 in such a way that the center of the 9 cells in Layer 1 is at [i,j] position and the other 8 cells are surrounding the middle one. The 9 cells are connected to a ***threshold decider*** that decides whether it is an edge or not. The ***threshold decider*** is performing the ***function*** f_{21} in (EQ 2.1) which maps to a binary value with 1 indicating an edge cell and 0 indicating a non-edge cell. Each connection from the 9 cells in Layer 1 to the threshold decider has different weighting. The corresponding weighting is represented by Matrix A (refer to chpater 3). Therefore the function between Layer 1 and Layer 2 is actually the convolution of Layer 1 by a mask and the resulting value is either 1 (edge) or 0 (non-edge) and assigned to Layer 2.

2.3.2 Interconnection and Functioning Between Layer 2 and Layer 3

The major function of the cells in Layer 3 is to record the features classified by the interlayer connection device between Layer 2 and Layer 3. The device is named as the ***attribute classifier***. The attribute classifier takes input the contents of the cells in Layer 2, if the cell at position [i,j] is an edge, it then checks if it is an end point or not, if it is not an end point, then tell the cell at [i,j] that it is an NULL feature else assign the feature attribute to the cell [i,j] in Layer 3. If there is only one edge at [i,j] in Layer 2 , then the feature classifier will treat it as a noise and simply discard it. If the cell at [i,j] is not an edge, then NULL edge is assigned. Figure 2.3 shows the interlayer connection between Layer 2 and Layer 3.

Figure 2.3 Interlayer Connection Between Layer 2 and Layer 3


Cells in Layer 2
Classification Window
If the middle of the classification window is an edge cell and is an end point, then the attribute classifier will classify the edge as an attribute and record by the cell in layer 3.
Attribute Classifier
Cells in Layer 3
Each cell in Layer 3 can record analog value from 0 to 255

For a 3x3 feature classifier mask arranged as a matrix B with the elements b_{xy} where x and y is an integer such that $x \in \{1,2,3\}$ and $y \in \{1,2,3\}$. Consider a cell at position [i,j] in Layer 2, if it is an edge cell, then cell [i,j] is defined as an non-end point if it satisfies the conditions in Table 2.1:

Table 2.1 The Conditions for a Cell in Layer2 to be a Non-end Point.

1. f_2(Layer2[i,j]) = Edge ; and
2. f_2(Layer2[i-1,j-1]) = f_2(Layer2[i+1,j+1]) = Edge ; xor
3. f_2(Layer2[i-1,j]) = f_2(Layer2[i+1,j]) = Edge ; xor
4. f_2(Layer2[i-1,j+1]) = f_2(Layer2[i+1,j-1]) = Edge ; xor
5. f_2(Layer2[i,j-1]) = f_2(Layer2[i,j+1]) = Edge.

Where for $f_2(z)$, $f_2 \rightarrow$ {Edge, Non-edge}, $z \in \{0,1\}$,

$f_2(z)$ = Edge if $z = 1$ and $f_2(z)$ = Non-edge if $z = 0$;

A cell in Layer 2 is said to be belong to a set ζ if it satisfies the conditions listed in Table 2.1. All elements in ζ are defined as non-end points for the cells in Layer 2. The function f_{32} in (EQ3) will sort out for the non-end points in Layer2. Now the cells in Layer 3 can be defined as

If Layer2[i,j] = 0 then Layer3[i,j] = NULL; else

$$\text{Layer3}[i,j] = f_{32}\left(\sum_{m=i-1}^{i+1} (b_{m-i+2,n-j+2}) * \text{Layer2}[m,n] \right) \text{......} \quad \text{(EQ 2.3)}$$

Where b_{xy} is an element of the attribute classifier mask B with $x \in \{1,2,3\}$ and $y \in \{1,2,3\}$;

B is a 3x3 matrix with b_{xy} as its elements which are defined as :

$$B = \begin{pmatrix} 2^0 & 2^1 & 2^2 \\ 2^3 & 0 & 2^4 \\ 2^5 & 2^6 & 2^7 \end{pmatrix}$$

f_{32} is a function that maps to an attribute value which is then recorded by the cells in Layer3. Now let a set $\Re$ which is defined as {0,24,36,66,129,255}, let $\wp$ be the set of integers ranging from 0 to 255, then f_{32} is defined as :

$$f_{32}(\gamma) = \text{NULL} \quad \text{if } \gamma \in \Re \text{ ; ......} \quad \textbf{(EQ 2.4)}$$

$$f_{32}(\gamma) = \gamma \quad \text{if } \gamma \in (\wp - \Re). \text{ ......} \quad \textbf{(EQ 2.5)}$$

Clearly, Function f_{32} will return a number ranging from 0 to 255 excluding all the members of $\Re$. In the computational point of view, one byte can be assigned for each element in Layer 3. To record the NULL attribute, we can arbitrarily assign one member in $\Re$ as the NULL attribute. This is because f_{32} will never return such a value. The design can be made in such a way that if a member of $\Re$ is present in Layer 3, then it is understood that it is a NULL attribute.

There is one more point to mention for the function f_{32}, if a cell at position [i,j] in Layer 2 is an edge, but f_{32} returns a value NULL according to (EQ 2.4). This is due to the fact that the value of

$$\sum_{m=i-1}^{i+1} \sum_{n=j-1}^{j+1} (b_{m-i+2,n-j+2}) * \text{Layer2}[m,n]$$

is 0. If we look closely to matrix B, the only possibility to produce this result is that matrix B is convoluting the 9 cells in Layer 2 with only the middle one an edge, all the surrounding 8 cells are non-edge cells. Since 0 is a member of $\Re$, a NULL attribute will be returned by function f_{32}. The reason is that the Image Analysis Architecture treats it as a noise and it is simply removed and ignored.

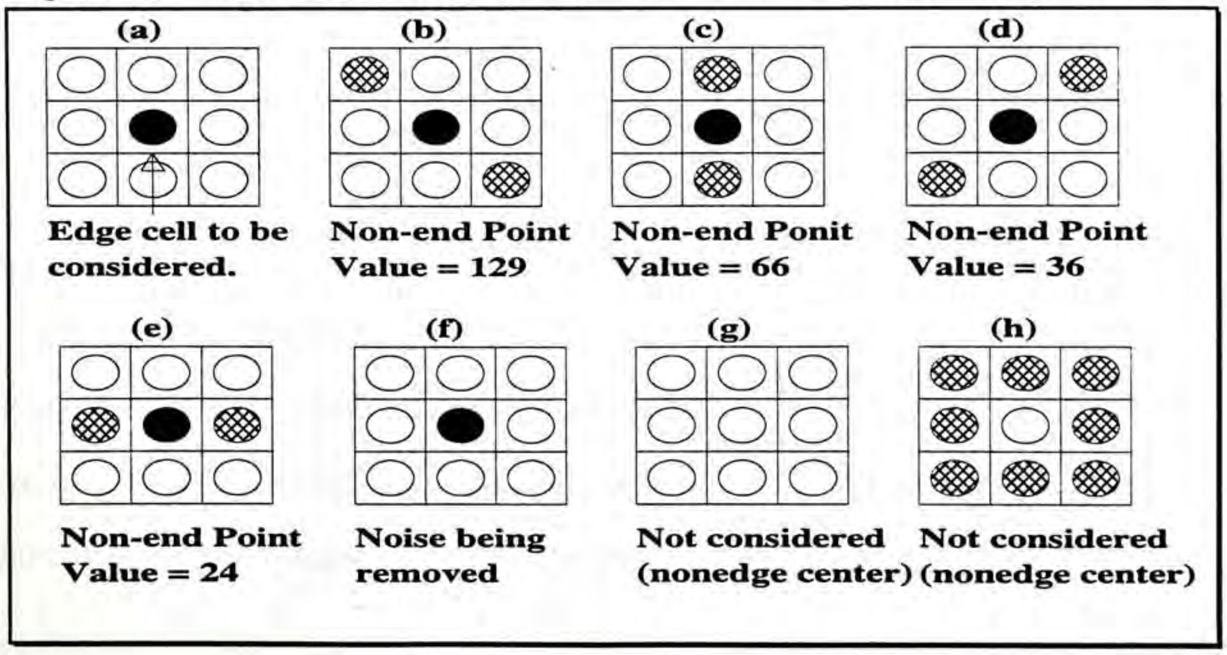
Figure 2.4 depicts the conditions listed in Table 2.1. A non-end point is actually a straight line with at least 3 pixels long that passes through the center of the convoluting attribute classifier mask which is a 3x3 matrix like matrix B. Figure 2.4 (a) shows that whenever we consider the attribute of the edge, only the center point is being taken into consideration. Figure 2.4 (b) illustrates a non-end point with attribute value 129, figure 2.4 (c) to figure 2.4 (e) are non-end points of attribute value 66, 36 and 24 respectively. Figure 2.4 (f) is considered as a noise. Since only the center of the attribute classifier mask is considered, both figure 2.4 (g) and figure 2.4 (h) will not be considered as an attribute.

The value returned from the function f_{32} can make sure that it is the unique value to represent the same edge pattern or attribute. It is obvious that for the same value, the edge pattern is the same. To show the uniqueness, arrange the elements of the matrix B as a binary bit map as shown below :

$$b_{33}\ b_{32}\ b_{31}\ b_{23}\ b_{21}\ b_{13}\ b_{12}\ b_{11}$$

The presence of an edge in position b_{xy} where $x \in \{1,2,3\}$ and $y \in \{1,2,3\}$, where x and y cannot be equal to 2 at the same time, is corresponding to a binary digit 1 at the position occupied by b_{xy}. Any equivalent edge pattern will produce the same bit pattern in one byte. Therefore the representation is unique for the same edge pattern.

Figure 2.4 The illustration of Non-end Point Feature Attributes.



2.3.3 Interconnection and Functioning Between Layer 3 and Layer 4

The major functions of Layer 4 is for the detection of moving objects. There are 2 stages for the cells in Layer 4 to work. In the first stage, the cells are initialized with the input from the results in Layer 3. In the second stage, the initialized cell in Layer 4 propagates the detection information to all of its neighbors with the verification from the gray level comparison results of the images stored in Layer 1.

Now we are about to consider the relationship between Layer 4 and Layer 3. As mentioned above, the cells in Layer 4 takes the input from Layer 3. The relationship can be best described by (EQ 2.6) below :

$$\text{Layer4}[dx,dy,i,j] = f_{43}(\text{Layer3}[i,j],\text{Layer3}[i+dx,j+dy]) \;\ldots\ldots\; \textbf{(EQ 2.6)}$$

Where Layer4[dx,dy,i,j] is the position [i,j] of [dx,dy] sublayer;

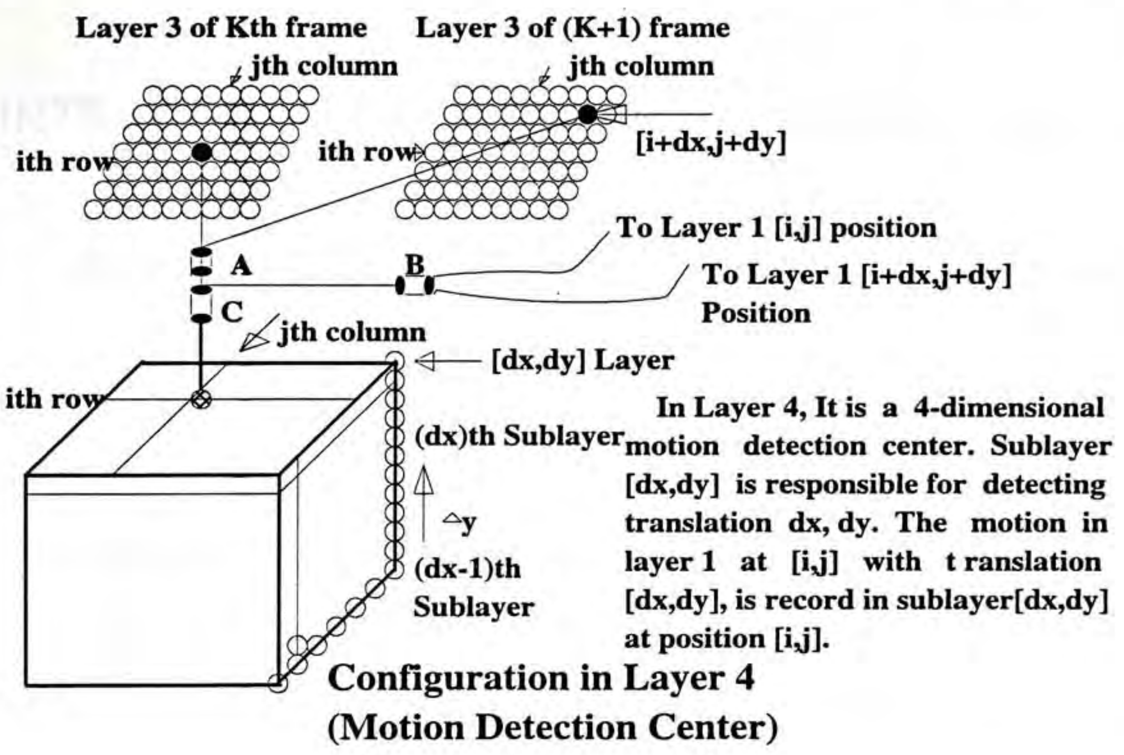
f_{43} is a function that takes 2 inputs z_1 and z_2, with

$z_1 \in (\wp - \Re)$, $z_2 \in (\wp - \Re)$ and $f_{43} \rightarrow \{0,1\}$.

(We only consider translation at this stage, so either 0 [no motion], or 1 [translation] can be returned by the function f_{43}.)

After all the cells in Layer 4 have been initialized by (EQ 2.6), the cells perform the second stage of action mentioned earlier. In the second stage, all the cells in Layer 4 will then perform subset illumination in order to minimize the errors of matching in feature level. Finally, the cells in level four will perform object integration and to build the motion correspondences with the input of the segmented image taken at time = $\mathbf{t_n}$. At the same time the shape of the objects being considered will be identified at the same time. These activities and the mathematical formulation will be discussed in Chapter 5 and Chapter 6.

Figure 4 Interconnection and Functioning Between Layer 4 and Layer 3.


Layer 3 of Kth frame
Layer 3 of (K+1) frame
jth column
jth column
ith row
ith row
[i+dx,j+dy]
To Layer 1 [i,j] position
A
B
To Layer 1 [i+dx,j+dy] Position
C
jth column
[dx,dy] Layer
ith row
In Layer 4, It is a 4-dimensional motion detection center. Sublayer [dx,dy] is responsible for detecting translation dx, dy. The motion in layer 1 at [i,j] with t ranslation [dx,dy], is record in sublayer[dx,dy] at position [i,j].
(dx)th Sublayer
△y
(dx-1)th Sublayer
Configuration in Layer 4
(Motion Detection Center)

Up to now, the brief discussion for translation detection is complete. In the following chapters, the mathematical formulation and the functions of the cells will be investigated in detailed. The experimental results and the discussions about this field will also be given.

CHAPTER 3

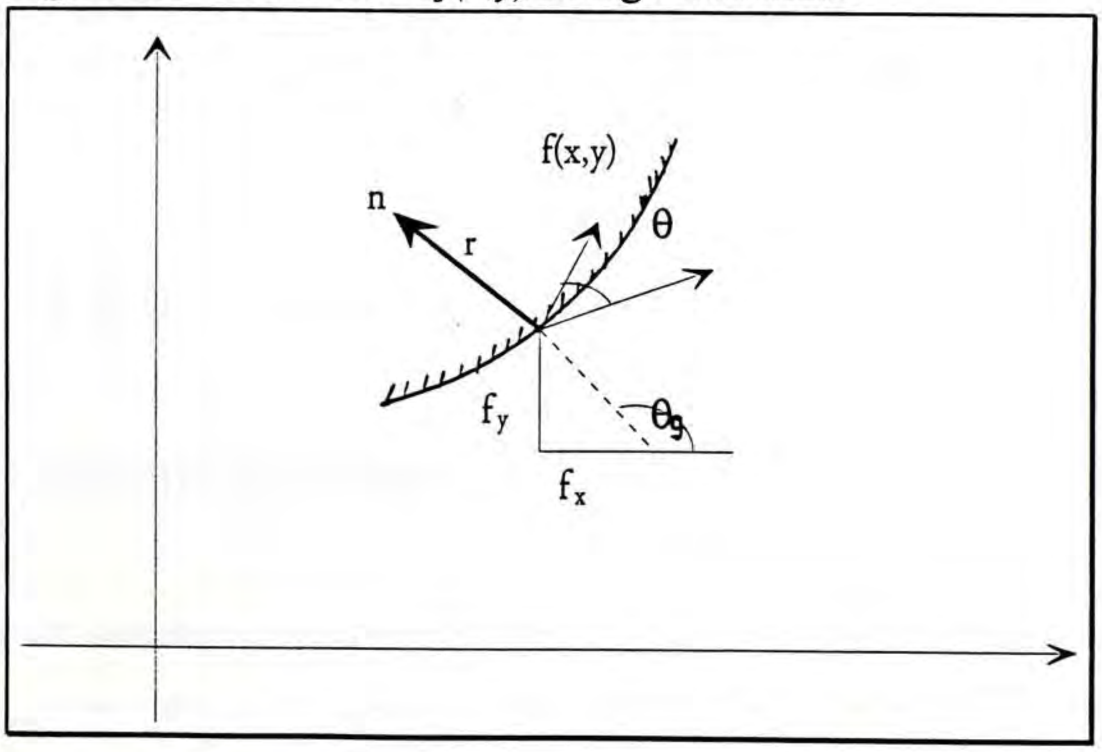
INTERCONNECTION BETWEEN LAYER1 AND LAYER 2

EDGE DETECTION AND INTENSITY GRADIENT

3.1 Introduction

In our proposed motion detection architecture, edge detection is performed in Layer 2. Edges characterize object boundaries and are therefore useful for segmentation, registration, and identification of objects in scenes. Edge points can be thought of as pixel locations of abrupt gray level change. For a continues image *f(x,y)*, its derivative assumes a local maximum in the direction of the edge. Therefore, one edge detection technique it to measure the gradient of *f* along *r* in a direction θ , Figure 3.1, that is :

Figure 3.1 Gradient of *f(x,y)* along *r* direction.



$$\frac{\partial f}{\partial r} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r} = f_x \cos\theta + f_y \sin\theta \quad \text{...... (EQ 3.1)}$$

The maximum value of $\partial f / \partial r$ is when $(\partial / \partial\theta)(\partial f / \partial r)$ is equal to 0. This gives

$$-f_x \sin\theta_g + f_y \cos\theta_g = 0$$

or

$$\theta_g = \tan^{-1}(\frac{f_y}{f_x}) \quad \text{...... (EQ 3.2)}$$

$$(\frac{\partial f}{\partial r})_{\max} = \sqrt{f_x^2 + f_y^2} \quad \text{...... (EQ 3.3)}$$

where θ_g is the direction of the edge. Based on this concepts, two types of edge direction operators have been introduced [Ref. 3.1 ~ Ref.3.6], *gradient operators* and *compass operators*. For digital images, these operators are also called masks which are representing finite difference approximations of either the orthogonal gradients f_x, f_y or the directional gradient $\partial f / \partial r$. Let **H** denote a *pxp* mask, for any image **U** , their inner product at location (m,n) is defined as

$$< U, H >= \sum_i \sum_j h(i,j)u(i+m, j+n) = u(m,n) \otimes h(-m,-n) \quad \text{...... (EQ 3.4)}$$

3.2 The gradient Operators

These are represented by a pair of masks H_1, H_2, which measure the gradient of an image *u(m,n)* in two orthogonal directions. The gradient vector magnitude *g(m,n)* can be defined as

$$g(m,n)=\sqrt{g_1^2(m,n)+g_2^2(m,n)} \quad \text{...... (EQ 3.5)}$$

$$\theta_g(m,n)=\tan^{-1}\frac{g_2(m,n)}{g_1(m,n)} \quad \text{...... (EQ 3.6)}$$

Where $g_1(m,n)=<U,H_1>$

$g_2(m,n)=<U,H_2>$

For approximation, the magnitude gradient is usually calculated as

$$g(m,n)=|g_1(m,n)|+|g_2(m,n)| \quad \text{...... (EQ 3.7)}$$

(EQ 3.7) is easy to perform and is preferred especially when the implementation is carried out in digital hardware circuits.

Table 3.1 is a list of the common gradient operators. The Prewitt, Sobel operators compute horizontal and vertical differences of local sums. This reduces the effect of noise in the data. These operators have the desirable property of yielding zeros for uniform regions.

Table 3.1 Some Common Gradient Operators. The bracket indicates the origin.

	H_1	H_2
Robert	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Smoothed Prewitt	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
Sobel	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Isotropic	$\begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$

The pixel location (m,n) is declared an edge location if g(m,n) exceeds some threshold *t*. The locations of edge points constitute an edge array $\varepsilon(m,n)$ which is defined as

$$\varepsilon(m,n) = \left\{ {}^{1,(m,n)\in I_g}_{0,otherwise} \right. \quad \text{...... (EQ 3.8)}$$

where

$$I_g = \{(m,n); g(m,n) > t\} \quad \text{...... (EQ 3.9)}$$

The edge map gives the necessary data for tracing the object boundaries in an image. Typically, *t* may be selected by using the accumulated histogram of g(m,n) so that 5 to 10% of pixels with largest gradients are declared as edge.

3.3 The Compass Operators

Compass operators measure gradients in some selected directions. Table 3.2 illustrates four different compass gradients for north-going edges. The shifting of the operators in anti clockwise direction gives a 45^o rotation.

Table 3.2 Compass Gradient Operators

$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$
$\begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 2 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Let $g_k(m,n)$ denote the compass gradient in the direction $\theta_k = \pi/2 + k\pi/4$ where k = 0, ..., 7. The gradient at the position (m,n) is defined as

$$g(m,n) = \max_k \{|g_k(m,n)|\} \text{...... (EQ 3.10)}$$

which can be thresholded to obtain the edge map as the gradient operators. Since only four of the compass gradients are linearly independent, it is possible to define four 3x3 arrays that are mutually orthogonal and span the space of these compass gradients. These arrays are called orthogonal gradients and can be used in place of the compass gradients [Ref. 3.7]. Compass gradient operators with higher angular resolution can be designed by increasing the size of the mask.

3.4 Laplace Operators and Zero Crossings

As the transition region gets wider, it is more advantageous to apply the second-order derivatives. One frequently encountered operator is the Laplacian operator which is defined as :

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \text{ ...... (EQ 3.11)}$$

Three of the different approximations of this operator are listed in Table 3.3. Because of the second order derivatives, this gradient operator can be more sensitive to noise than those of the previously defined ones.

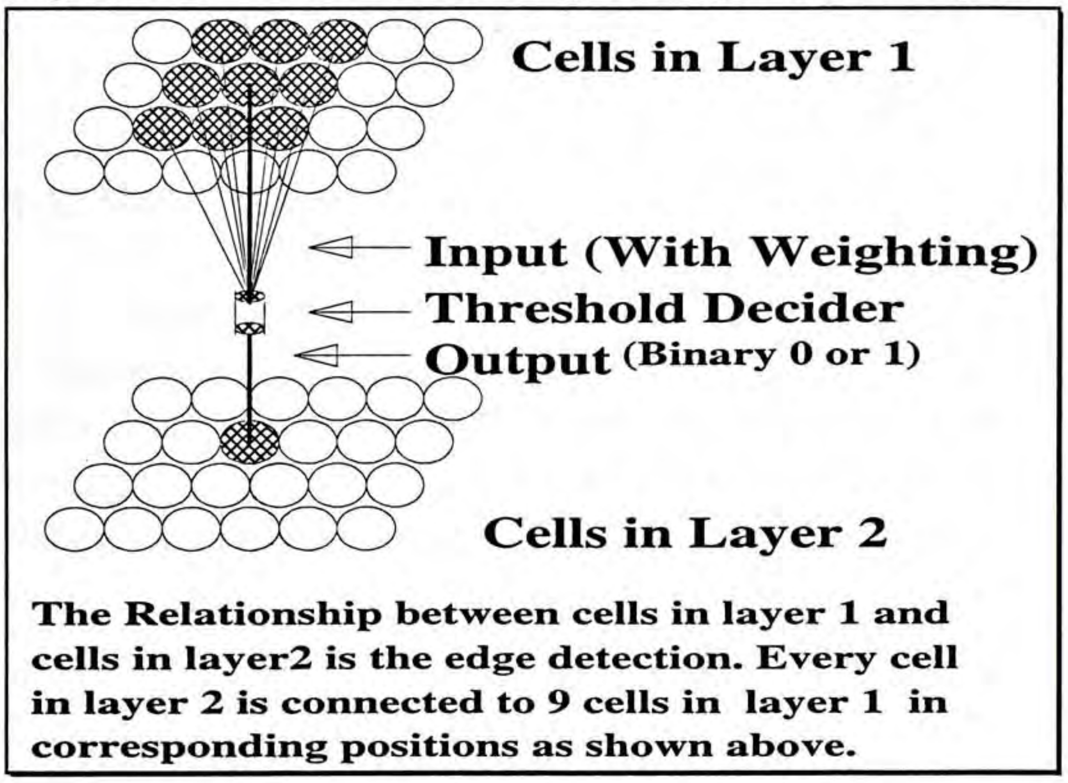
Table 3.3 Discrete Laplace Operators

$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$

3.5 The Connection Between Layer 1 and Layer 2

The connection between layer 1 and layer 2 in the proposed architecture can now be clearly illustrated in the Figure 3.2 when we use a 3x3 mask for edge detection:

Figure 3.2 The Connectivity of Layer 1 and Layer 2.



The Relationship between cells in layer 1 and cells in layer2 is the edge detection. Every cell in layer 2 is connected to 9 cells in layer 1 in corresponding positions as shown above.

For a cell of layer 2 at position (m,n), connections are made to the cell of layer 1 at position (m,n) and also all the immediate neighbor cells of (m,n) of layer 1. The weightings for each connection may not be the same but depends on the selection of the mask. The elements of the selected mask are actually the corresponding weightings for the connections in the relative positions.

CHAPTER 4

TECHNIQUES FOR FEATURE CLASSIFICATION AND BUILDING FEATURE CORRESPONDENCES

4.1. Introduction

In our pixel based part of motion detection architecture shown in Figure 1.1a, it first records the images obtained from an image digitizer in layer 1. Then edge detection is executed and the result is recorded in layer2. From the edges detected, reorganization are performed to sort the image elements into feature classes. Then features of two consecutive images are compared to develop the correspondences.

Layer3 is dedicated for feature classification. Edge pixels are encoded to become integers used in the matching algorithm in layer 4. The relationship between layer 3 ($L_3[i,j]$) and layer 2 ($L_2[i,j]$) can be described by **(EQ 2.3)** :

$$L_3[i,j] = f_{32a}(f_{32b}(L_2[i,j])) \quad \ldots\ldots \quad \text{(EQ 2.3)}$$

Where the feature classification functions f_{32a} ( )and f_{32b} ( ) will be further defined in Section 4.3.

The results of edge detection performed by (EQ 2.3) are binary, i.e. either o or 1. If these edge points are used to build object correspondences, the spatial information of the edges will be lost. Consequently, the chance of mismatch is greatly increased. Other researchers have attempted the problem before, for example , according to Barsky and Greenberg (1980) they used B-spline to represent the edges , and for Person and Fu (1977) Fourier Descriptors were used. However, these methods have their disadvantages in common. (1) They are only approximate representations, (2) they use intensive floating point calculation, (3) they have difficulties for matching similar features and (4) the operations are complex and time consuming. To overcome the above problems, we proposed a new method to represent the local spatial information of an edge point. This representation , which is coded as integers, is simple, and no floating point calculation involved.

The classification scheme is carried out by functions f_{32a} and f_{32b} which will be discussed here.

Let **B** be an **m x m** matrix where **m** is equal to 2k+1 with k is an integer greater than 1. Let $\mathbf{b}_{xy}$ be an element of matrix **B** at row x and column y, then all members of the set

$$\boldsymbol{e} = \{\mathbf{b}_{xy} : \text{for all } \mathbf{x} \in \{0,1,...,m-1\} \text{ and all } \mathbf{y} \in \{0,1,...,m-1\}\}$$

are the elements of the matrix **B**.

Lemma 1 : If there are **n** possible events which can happen independently, then a positive integer ranging from **0** to $\mathbf{2^n-1}$ is sufficient to indicate which of the n events have happened and any value in this range is giving a unique representation of which of the **n** events have happened.

Proof : Arrange the **n** events as event **0, 1, 2, ... , n-1.** Since the events are either happening or non-happening, then a binary value is sufficient to tell that the representing event has either happened or not happened. Suppose there are n binary bits named as $\mathbf{b}_0$, $\mathbf{b}_1$, $\mathbf{b}_2$, ..., $\mathbf{b}_{n-1}$ respectively, a value of 1 in $\mathbf{b}_k$, where $\mathbf{k} \in \{0, 1, 2, ..., n-1\}$, means that event **k** has happened; a value of 0 means that event **k** does not happen. If all the events have happened, then all the **n-1** bits are set, this is equivalent to an integer $\mathbf{2^n-1}$; if all **n-1** bits are not set, this is equivalent to an integer **0**. But each bit is used to represent an independent event, thus the representation is unique and hence Lemma 1 is proven.

Lemma 2 : For an **m x m** matrix **B**, if all elements are binary values only, then a positive integer ranging from **0** to $\mathbf{2^{m \times m}-1}$ is sufficient to represent all properties of **B**.

Proof : Let matrix **B** be

$$\begin{pmatrix} b_{0\,0} & b_{0\,1} & \ldots\ldots & b_{0\,m-1} \\ b_{1\,0} & b_{1\,1} & \ldots\ldots & b_{1\,m-1} \\ & & \ldots\ldots & \\ b_{m-1\,0} & b_{m-1\,1} & \ldots\ldots & b_{m-1\,m-1} \end{pmatrix}$$

let $\mathbf{k = i*m + j}$ where $\mathbf{i} \in \{\mathbf{0, 1, 2, \ldots\ldots, m\text{-}1}\}$ and
$\mathbf{j} \in \{\mathbf{0, 1, 2, \ldots\ldots, m\text{-}1}\}$.

Then the elements of **B** can be arranged as $\mathbf{b_0, b_1, \ldots, b_k, \ldots, b_{mxm-1}}$.

1. The positional representation is unique. i.e. both **i** and **j** can be uniquely defined with

$$\mathbf{j = k \bmod m} \text{ and } \mathbf{i = (k-j)/m}.$$

2. Assign

$$\mathbf{b_k} = \mathbf{b_{ij}} \ldots\ldots \text{(EQ 4.1)}$$

where $\mathbf{i} \in \{\mathbf{0, 1, 2, \ldots\ldots, m\text{-}1}\}$ and
$\mathbf{j} \in \{\mathbf{0, 1, 2, \ldots\ldots, m\text{-}1}\}$,

then if all bits is set, the equivalent decimal integer value is $\mathbf{2^{(m-1)*(m-1)}-1}$ and this is the largest value. By **Lemma 1**, the integer value can be broken into binary patterns of **m*m** bits which is equivalent to the elements of **B** and therefore **Lemma 2** is proven. Hence there exists a function such that

$$\mathbf{B} = f_t(\mathbf{b_0, b_1, \ldots, b_k, \ldots, b_{mxm-1}}) \ldots\ldots \text{(EQ 4.2)}$$

Suppose that there is an edge array $\boldsymbol{E}$ of size **n x n**, the value of each edge is binary with 1 indicating an edge and 0 indicating a non-edge. For a window $\boldsymbol{W}$ of size **m x m** which is completely included in $\boldsymbol{E}$, by using **Lemma 1** and **Lemma 2**, there exists an integer ranging from 0 to $\mathbf{2^{mxm}-1}$ to represent the position of the edge element(s) in the window $\boldsymbol{W}$.

Let us define a classification of matrix **B** as

$$\mathbf{B} = \begin{pmatrix} 2^0 & 2^1 & \ldots\ldots & 2^{m-1} \\ 2^m & 2^{m+1} & \ldots & 2^{2m-1} \\ & \ldots\ldots & & \\ 2^{m(m-1)} & & \ldots\ldots & 2^{mm-1} \end{pmatrix} \ldots\ldots \text{(EQ 4.3)}$$

with $\mathbf{b}_{ll} = \mathbf{0}$ where $\boldsymbol{l} = \mathbf{(m+1)/2}$.

If B is a 3 x 3 matrix, by using Lemma 1 and 2, the edge feature of an edge in L2[i,j] can be transformed into an integer by f_{32b} ( ).

$$f_{32b}(L_2[i,j]) = \sum_{m=i-1}^{i+1} \sum_{n=j-1}^{j+1} (b_{m-i+2,n-j+2}) * L_2[m,n]) \ldots\ldots \text{(EQ4.4)}$$

Where $\mathbf{b}_{xy}$ is an element of matrix **B** at row **x** and **y** column.

The results of $L_2[i,j]$ is called a ***spatial feature***. Some spatial features are too common and will not be considered as edge features, let the set of integers which will ***not*** be considered as edge features be $\boldsymbol{n}$, and let the set of integers of all possible features be $\boldsymbol{p}$, then the set z of edge features is defined as :

$$z = \boldsymbol{p} - \boldsymbol{n} \ldots\ldots \text{(EQ 4.5)}$$

Function f_{32a} is called a ***selective function*** which is defined as :

$$f_{32a}(w) \quad \begin{aligned} &= w && \text{if } w \in z \\ &= 0 && \text{if } w \in \boldsymbol{n} \end{aligned} \quad \ldots\ldots \text{(EQ 4.6)}$$

Here, it is time to clarify the terms used in this section, matrix **B** is called a ***classification matrix***, functions f_{32a} and f_{32b} are collectively called ***feature classification functions*** or a ***feature classifier***. A member of z is called ***an edge attribute***, a member of $\boldsymbol{n}$ is called a ***non-edge attribute***.

To illustrate these techniques, suppose matrix **B** is a **3 x 3** matrix, by using **(EQ4.2)**, it is defined as :

$$\mathbf{B} = \begin{pmatrix} 2^0 & 2^1 & 2^2 \\ 2^3 & 0 & 2^4 \\ 2^5 & 2^6 & 2^7 \end{pmatrix}.$$

Since we only consider the points where the central element is an edge point, then the central element of matrix **B** can be 0 to save one bit. For those points where the central element is not 1, we do not consider it as a feature.
It is clear that :

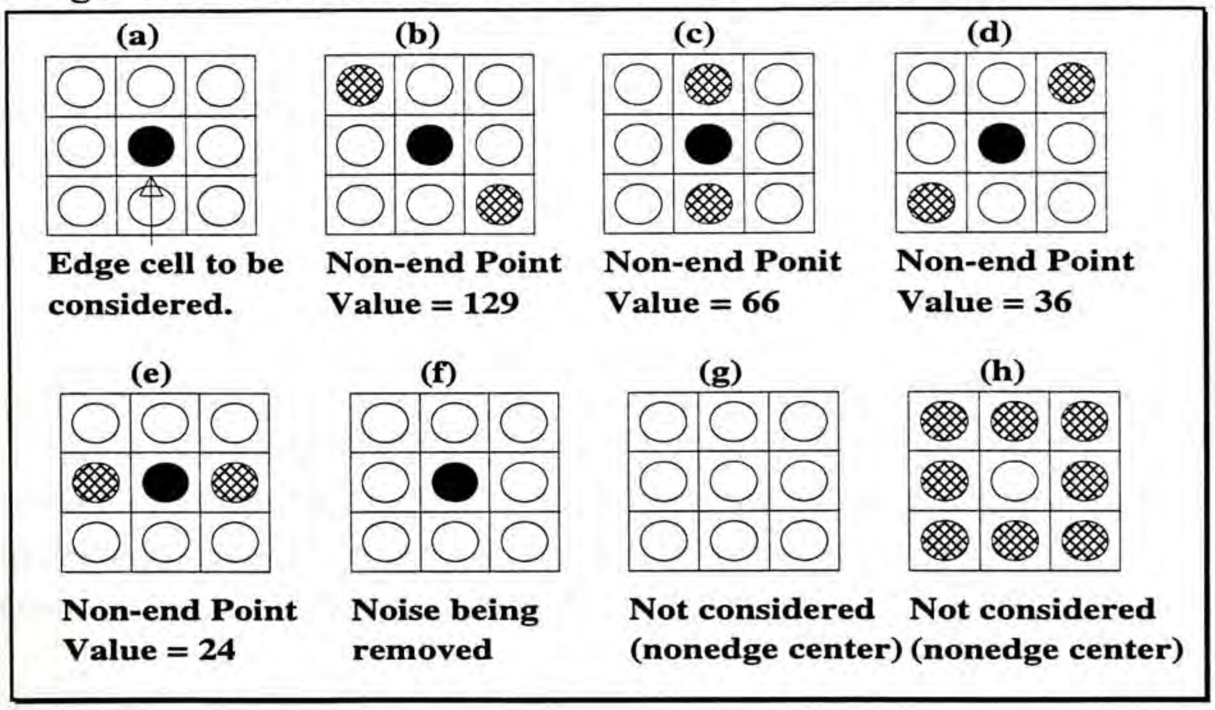
$$p = \{ 0, 1, 2, 3, \ldots\ldots, 255 \}$$

Let $n = \{ 0, 24, 36, 66, 129, 255 \}$
and $z = p - n$

The set of non-edge attributes is illustrated by Figure 4.1. These members of the set ***n*** are usually the straight lines which are passing through the center of the matrix **B** or only one stand alone edge point that is considered as noise.

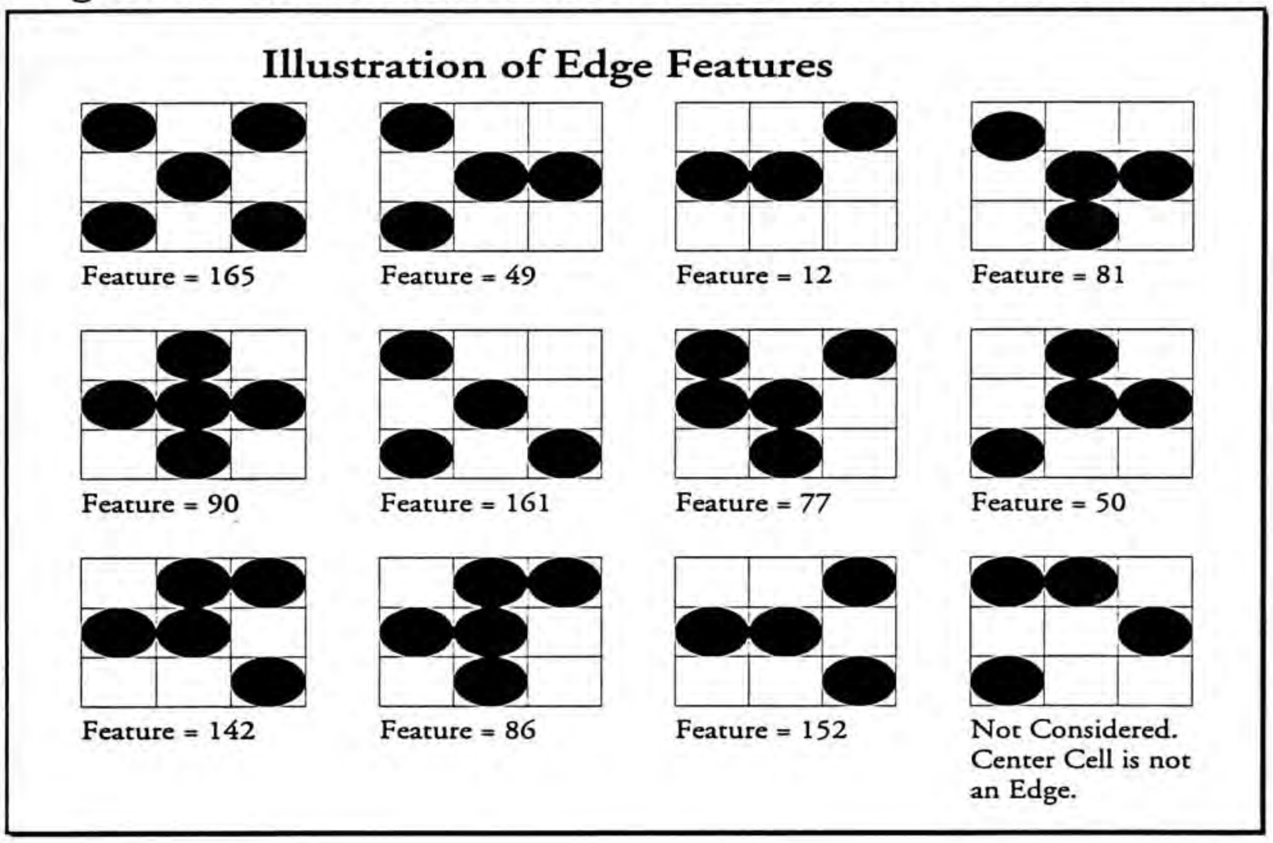
According to this set of data, Figure 4.2 illustrates how the classification matrix is used to classify the edge information in the edge array. The result is represented as an integer ranging from 0 to 255. It is obtained by the convolution of the feature classification matrix with the edge information array in layer 2. The results are recorded in layer 3 of the image analysis architecture and will be further processed in layer 4 for attribute matching.

Figure 4.1 Illustrations of Non-end Point Feature Attributes.



As shown in Figure 4.2, the feature classifier will classify the edge features only if the center cell is an edge and at least one edge cell exists in the surrounding. The feature number is then recorded in the corresponding cells in layer 3.

Figure 4.2 Illustrations of some results of Feature Classification



4.2 Attribute Matching from Feature Classification

Once the edge features are classified and recorded in layer 3, feature matching can be carried out in layer 4. As introduced in Section B, layer 4 is a four dimensional layer for calculating and storing motion results. The detection of the translation of the object is performed in sub-layers of layer 4. For each element in layer 4, the value is governed by (EQ 4.7).

$$L_4[dx,dy][i,j] = f_{43}(L_3[i,j], R_3[dx+i,dy+j]) \text{ .... } \quad \text{(EQ 4.7)}$$

Function f_{43b} takes 2 parameters as the input, the main purpose is to measure the similarity of the edge patterns. From EQ. 8, the edges are transformed into their corresponding feature attributes. In our matching algorithm, the similarity of two feature attributes is measured by the following equation.

$$f_{43}(a,b) = \textbf{Number of bits set (a EXCLUSIVE-OR b)} \text{ ...... } \quad \text{(EQ4.8)}$$

This is a measure of how different the feature attribute **a** and **b are**. The calculation is simple and may be implemented by special hardware to gain speed. It also implicitly includes the function of inexact matching so that noise can be tolerated.

The likelihood of two features is inversely proportional to the output of (EQ 4.8), a value of "0" returned by the function f_{43a} have the maximum degree of matching, "255" is the minimum. The results returned by the function f_{43a} will be recorded by the corresponding sub-layers of layer4. To find the motion vector of the object being viewed we use the following searching method. For a pixel position of (i,j), we search for the minimum value in all the sub-layer elements $L_4[dx',dy'][i,j]$ for $dx'=0$ to dx-1 and $dy'=0$ to dy-1. The motion vector of pixel (i,j) is dx'' and dy'' if the sublayer with dx'',dy'' contains the minimum value.

4.3. Intermediate Results (just after feature matching)

Figure 4.3a shows a picture of an integrated circuit (I.C.) moving at a time interval of t_n (Resolution = 256 x 256 with 256 gray levels), the I.C. is moving in the horizontal (x) and vertical (y) directions. The picture of the moving object at t_{n+1} is in Figure 4.3b.

Figure 4.3a. Moving IC at time = t_n.

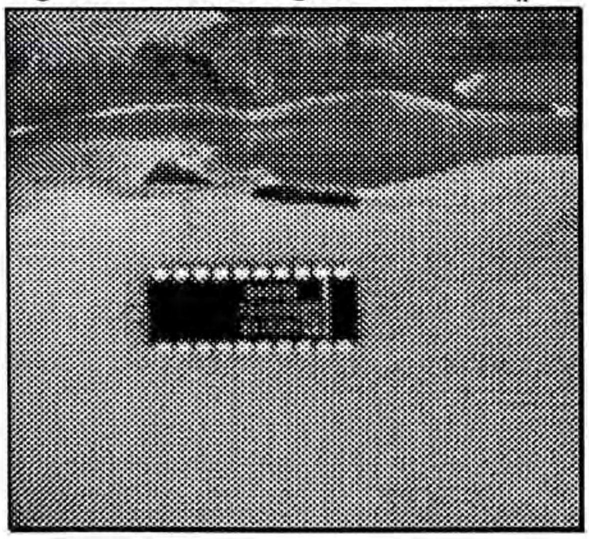
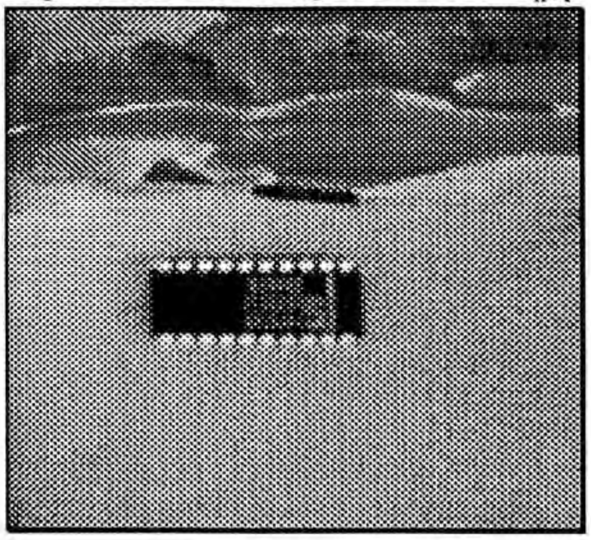


Figure 4.3b. Moving IC at time = t_{n+1}.



The sub-layers of layer4 are as shown by Figure 4.4a to Figure 4.4f . From the information of Figure 4.4a to Figure 4.4f , we can deduce the motion of the IC is either [3,2] (i.e. x=3, y=2) or [4,2] (x=4,y=2). The motion vector can be made as shown by Figure 4.5. Since this is the raw matching result, there may be some ambiguous results like Figure 4.4b and Figure 4.4e, the improvement and the removal of the ambiguous results will be discussed in Chapter 6.

Figure 4.4a Sublayer $L_4[3,1]$

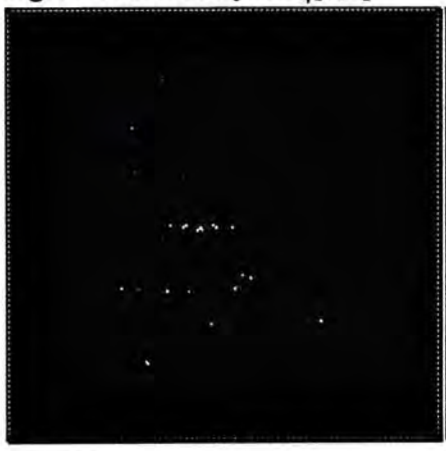


Figure 4.4b Sublayer $L_4[3,2]$

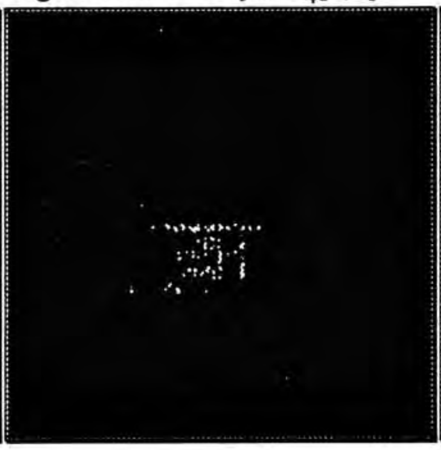


Figure 4.4c Sublayer $L_4[3,3]$

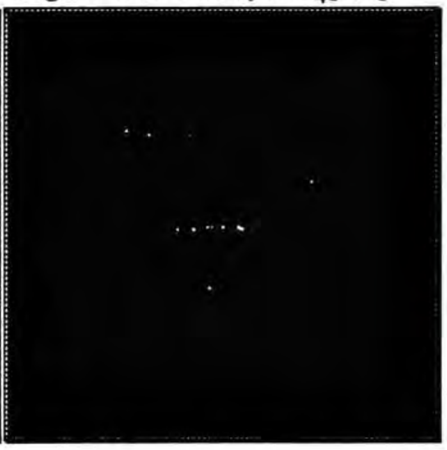


Figure 4.4d Sublayer $L_4[4,1]$

Figure 4.4e Sublayer $L_4[4,2]$

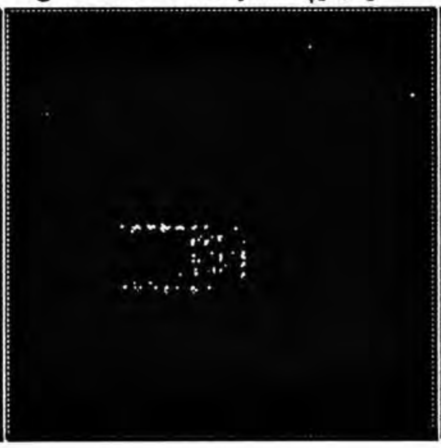
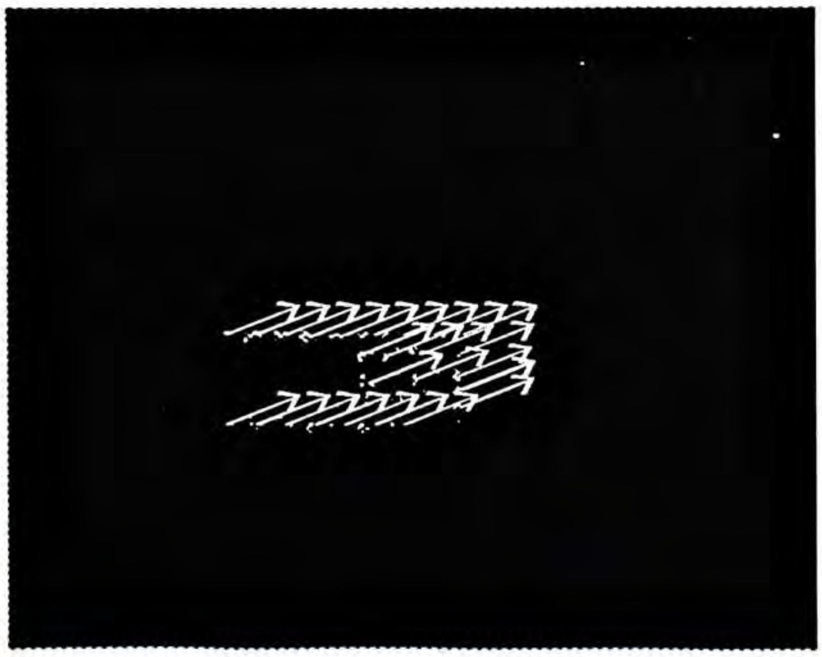


Figure 4.4f Sublayer $L_4[4,3]$

Figure 4.5 The motion vectors of the moving object .



CHAPTER 5

EDGE DIRECTED IMAGE SEGMENTATION

5.1. Introduction

The ultimate goal of image processing applications is to extract important features from the image data provided, from which a description, interpretation or may be the understanding of the scene can be provided by the image processing machines. For example, a vision system may be able to distinguish parts on an assembly line and list their features, such as their size and number of holes, an automatic traveling vehicle may be able to navigate and avoid collision by itself, etc..

The building of a successful image understanding system may involve image preprocessing, feature extraction, image segmentation, feature classification or description, feature matching, building feature correspondence table and finally the understanding of the scene provided by the image.

For the image preprocessing, we mean the removal of header, the techniques of image enhancement such as high pass, low pass, histogram equalization, sharpening, noise removal, etc.. The main purpose of image preprocessing is to improve the quality of the image for further processing. It, in fact, is not part of the image understanding system. There is a number of techniques which has been proven to work reasonably well.

Starting from feature extraction, some information can be extracted from the given image. The most common technique applied here is edge detection. There are many well known operators which have been developed by the early researcher and are proven to be sound. Further discussion is skipped here.

For feature classification, a sound technique has been developed earlier and is very efficient for computer application and feature matching[Chapter 4]. The main purpose of the technique mentioned in [Chapter 4] is take the spatial information into consideration. The result obtained by the application of [Chapter 4] is more accurate because of the consideration of spatial information. The matching technique is straight forward and efficient, or uses an **EXCLUSIVE OR** only, and this operator is very suitable for digital computer application.

Another branch of image processing, which is extremely important, is image segmentation. There is no acceptable technique which is developed so far though many researchers claim that their techniques are very good. In this paper, we will show how important it is and purpose a new technique which may obtain a better result.

Finally, matching can be carried out and the building of feature correspondences can be proceeded. The final goal of this paper is to build the object correspondence table and a motion map. The importance of image segmentation is to make the final decision that which is correct matching and which is mismatching.

In this chaper, a technique called **Edge Directed Image Segmentation** is proposed. An algorithm for **Visiting Homogeneous Region** is discussed. The **Matching Strength Calculation** is also defined. The **Techniques for Object Conflict Resolution and Translation Detection** is also described. In the process of building the Object Correspondence, a motion map is obtained. From this motion map, the motion information of the objects in the image array can be deduced.

5.2 Current Approaches to Image Segmentation

Basically, image segmentation refers to the decomposition of a scene into the components. It is a key step in image understanding. There are several approaches to this problem. The major and most common ones are being discussed in the following.

5.2.1 Amplitude Thresholding or Window Slicing

This technique is used when the amplitude features are sufficient to characterize the objects. The appropriate amplitude feature values are calibrated in such a way that a given amplitude interval represents unique object characteristic. This technique is also useful for the segmentation of binary images such as printed documents, line drawings and graphics, X-ray images, and so on.

The ways for selecting the threshold is an important step in this method. Some of the commonly used approached are as follows ;

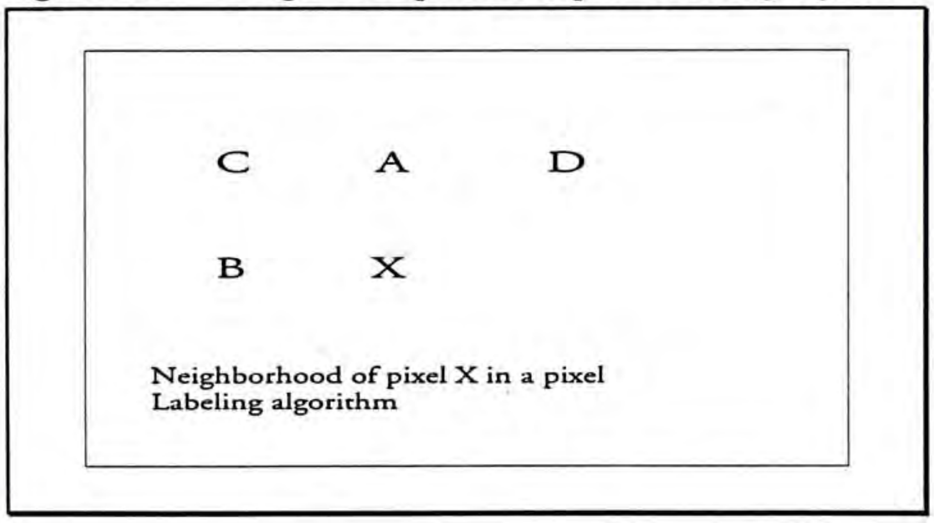
1. The histogram of the image is examined for locating peaks and valleys. If it is multimodal, then the valleys can be used for selecting thresholds.
2. Select the threshold (*t*) so that a predetermined fraction (*f*) of the total number of samples is below *t*.
3. Adaptive selection of threshold by examining histograms of local neighborhood.
4. Selectively threshold by examining histograms only of those points that satisfy a chosen criterion. For example, in low contrast images, the histogram of those pixels whose values are greater than a certain magnitude will exhibit a special feature than that of the original image.
5. If a probabilistic model of the different segmentation classes is known, determine the threshold to minimize the probability of error or some other quantity.

5.2.2 Component Labeling

A simple and effective method of segmentation of binary images is by examining the connectivity of pixels with their neighbors and labeling the connected sets. Two practical algorithms are as follows.

Pixel Labeling. Suppose a binary image is raster scanned left to right and top and bottom. The current pixel, X (Figure 5.1), is labeled as belonging to either an object (1s) or a hole (0s) by examining the connectivity to the neighbors A, B, C and D. For example, if X=1, the it is assigned to the object(s) to which it is connected. If there are two or more qualified objects, then those objects are declared to be equivalent and are merged. A new object label is assigned when a transition from 0s to an isolated 1 is detected. Once the pixel is labeled, the features of that object are objected. at the end of scan, features such as cencord, area, and perimeter are saved for each region of connected 1s.

Figure 5.1 Neighbor of pixel X in pixel labeling algorithm.

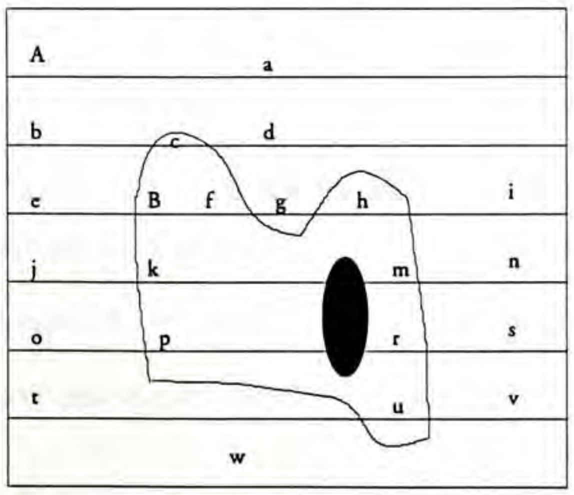


5.2.3 Run-length connectivity analysis.

An alternate method of segmenting binary images is to analyze the connectivity of run lengths from successive scan lines. To illustrate this idea, we consider Figure 5.2, where the black or white runs are denoted by a, b, c, ... A segmentation table is created , where the run a is named A. The first run of the next scan line, b, is of the same color as a and overlaps a. Hence b belong to the object A and is place underneath a in the first column. Since c is of different color, it is placed in a new column, for an object labeled B. The run d is of the same color as a and overlaps a. Since b and d both overlap a, divergence is said to have occurred, and a new column of object A is created, where d is placed. A divergence flag ID1 is set in this column to indicate the object B has caused this divergence. Also the flag ID2 of B (column 2) is set to A to indicate B has caused divergence in A. Similarly, convergence occurs when two or more runs of 0s and 1s in a given line overlap with a run of same color in the previous line. Thus convergence occurs in run u, which sets the convergence flags IC1 to C in column 4 and IC2 to B in column 6. similarly, w sets the convergence flag IC2 to A in column 2, and the column 5 is labeled as belonging to object A.

In this manner, all the objects with different closed boundaries are segmented in a single pass. The segmentation table gives the data relevant to each object. The convergence and divergence flags also give the hierarchy structure of the object. Since B causes divergence as well as convergence in a and C has a similar relationship with B, the objects A, B, and C are assigned levels 1, 2 and 3 respectively.

Figure 5.2 Run-length connectivity algorithm.



(a) Input : Binary image

Column	1	2	3	4	5	6
Level	1	2	1	2	1	3
Object	A	B	A	B	A	C
IC1			B	C		
ID1			B	C		
IC2		A				B
ID2		A				A
	a					
	b	c	d			
	e	f	g	h	i	
	j	k		m	n	l
	o	p		r	s	q
	t	u			v	
	w					

(b) Output: Segmentation table

5.2.4 Boundary-Based Approaches

Boundary extraction techniques segment objects on the basis of their profiles. Thus contour following, connectivity, edge linking and graph searching, curve fitting, Hough transformation, and other techniques are applicable to image segmentation. The difficulties with boundary-based methods occur when objects are touching or overlapping or if a break occurs in the boundary due to noise or artifacts in the image.

5.2.5 Region-Based Approaches and Clustering

The main idea in region-based segmentation techniques is to identify various regions in an image that have similar features. Clustering techniques encountered in pattern recognition issues have similar objectives and can be applied for image segmentation.

One class of region-based techniques involves *region growing* [Ref. 5.8]. The image is divided into atomic regions of constant gray levels. Similar adjacent regions are merged sequentially until the adjacent regions become sufficiently different as shown by Figure 5.3. The track lies in selecting the criterion for merging. Some merging heuristics are as follows:

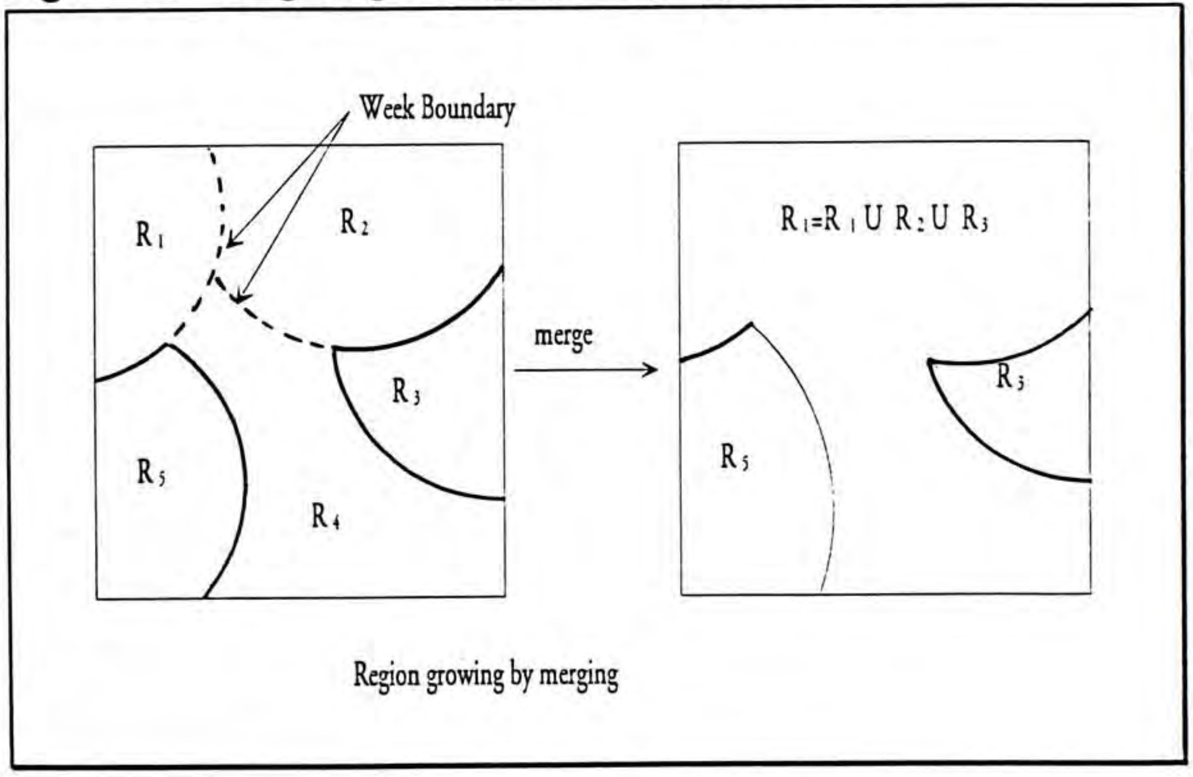
1. Merge two regions R_i and R_j if $w/P_m > O_l$, where $P_m = \min(P_i, P_j)$, P_i and P_j are the perimeters of R_i and R_j , and w is the number of weak boundary locations (pixels on either side have their magnitude difference less than some threshold s). The perimeter O_l controls the size of the region to be merged. For instance, $O_l = 1$ implies two regions will be merged only if one of the regions almost surrounds the other. Typically $O_l = 0.5$.

2. Merge R_i and R_j if $w/I > O_2$, where I is the length of the common boundary between the two regions. Typically O_2=0.75. so the two regions are merged if the boundary is sufficiently weak. Often this step is applied after the first heuristic has been used to reduce the number of regions.

3. Merge R_i and R_j only if there are no strong edge points between them. Note that the run length connectivity method for binary images can be interpreted as an example of this heuristic.

4. Merge R_i and R_j if their similarity distance is less than a threshold.

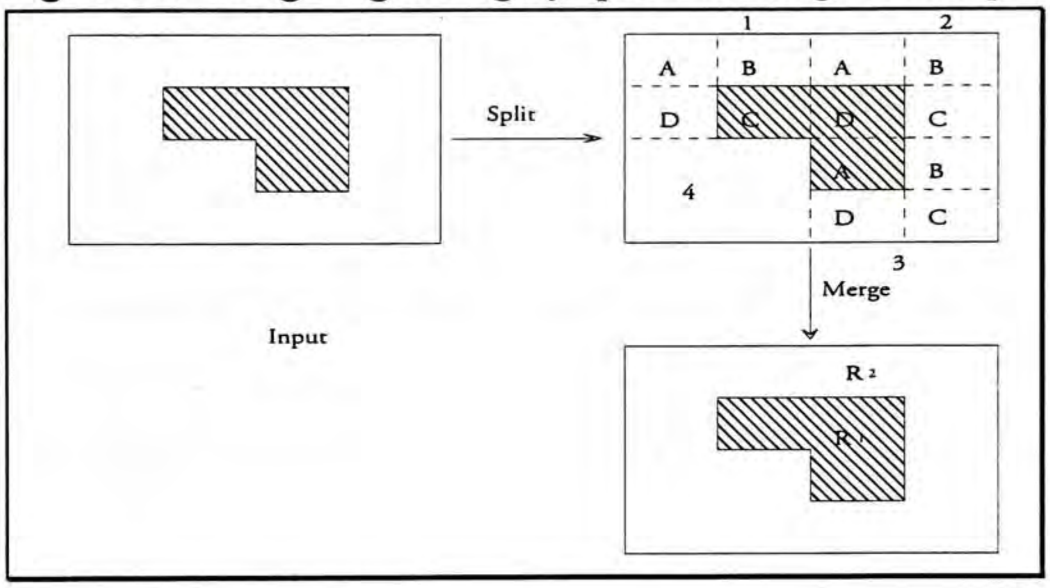
Figure 5.3 Region growing by merging illustration.



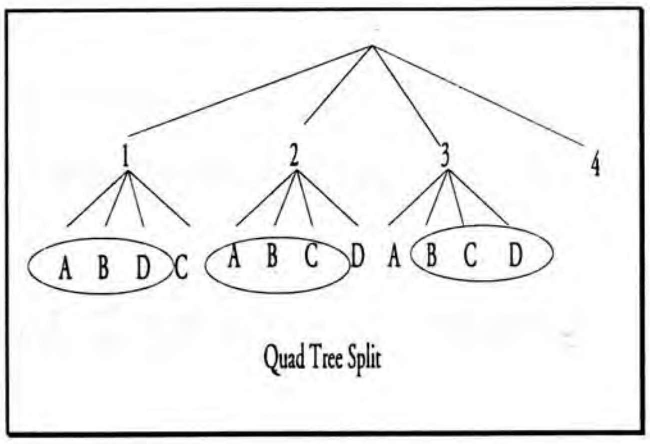
Instead of merging regions, we can approach the segmentation problem by splitting a given region. For example the image could be split by the *quad-tree* approach and then similar regions could be merged, as shown by Figure 5.4.

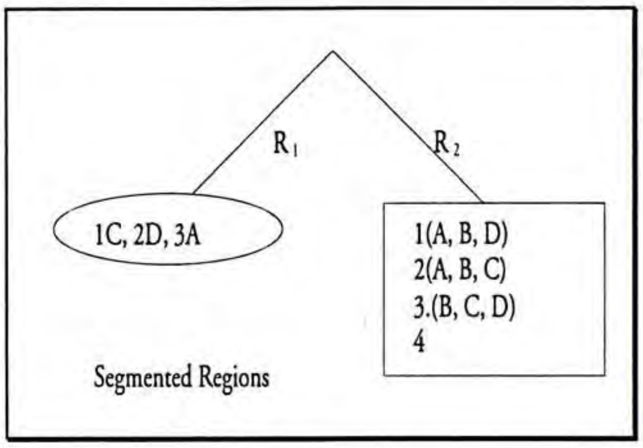
Region-based approaches are generally less sensitive to noise than the boundary-based methods but their implementation complexity can be quite large.

Figure 5.4 Region growing by split and merge techniques.



(a) Input





(b) Quad Tree Split (c) Segment regions

5.2.6 Template Matching

One direct method of segmenting an image is to match it against templates from a given list. The detected objects can then be segmented out and the remaining image can be analyzed by other techniques. This method can be used to segment busy images, such as journal pages containing text and graphics. The text can be segmented by template- matching techniques and graphics can be analyzed by boundary following algorithms.

5.2.7 Texture Segmentation

Texture segmentation becomes important when objects in a scene have a textured background. Since texture often contains a high density of edges, boundary based techniques may become ineffective unless texture is filtered out. Clustering and region based approach applied to textured features can be used to segment textured regions. In general, texture classification and segmentation is quite a difficult problem.

However, all the above mentioned techniques have their limitations which are suitable for a certain kind of image respectively, therefore is not suitable for general image analysis application. In next section, a new technique called **Edge Directed Image Segmentation** will be introduced for image analysis purpose.

5.3 Edge Directed Image Segmentation (used in our system)

As mentioned earlier, image segmentation is one of the basic problem in computer vision. The importance and the related applications have been discussed in Section A. The aim of the segmentation problem is to partition an image into regions according to some relationships. An attempt is made to find outlines of regions that would correspond to those places in the image across with the relationships break. The have been basically two approaches to this problem.

The first, the edge detection and boundary following approach [Ref. 5.2], which searches for discontinuities with respect to some picture function characteristics. The second, the region growing approach [Ref. 5.3-5.5], attempts to merge pixels of similar characteristics, especially the intensity (gray level) and the color attributes. Such a group of pixels form a region, the boundary of which is then the sought outline. The split and merge algorithm in [Ref. 5.6] and the pyramid-type algorithm in [Ref. 5.7] are based on this approach.

The emphasis in this paper is to take the advantages of the above mentioned 2 approaches and combine the results as one. The algorithm takes the input of the original image and the edge map of it. The output is an image with segmented regions.

Now, let me explain the importance of image segmentation in this project. After the matching of features in layer4, although the features can be identified by human eyes, there is no way for a machine to understanding which object the features are belong to. Despite the raw matching results can be improved by the **Subset Elimination** algorithm described in [1], the results are still ambiguous. Since the algorithm is applied in the pixel level, there is still a materially large quantity of mismatch information cannot be eliminated. These matching errors can only be eliminated in the object level. The completion of image segmentation process can enable the machine to recognize the image region of an object.

A new, simple and effective algorithm has been developed to handle the image segmentation problem. The algorithm takes the input of the **original image map**, **edge map of the original image** and a value of merging control, named as **Epsilon**. This algorithm need not trace the edge boundary, so the problem of breaking boundary is solved automatically. It is simply based on the following assumption.

There is at least one edge point detected on the boundary of an object.

This assumption is simple and extremely reasonable, almost all objects can satisfy this simple assumption. In fact, the boundary of an object must be an edge. The edge detection algorithms can be applied to obtain the edges of an image map.

The algorithm takes the inputs of the image map I_1 , the edge map of the image map E_1 and the input of a parameter d (Epsilon). **Algorithm 5.1** gives the detailed description of the algorithm. The algorithm first builds up a stack Q_1 with its elements be the order pairs of the coordinates of edge points of the image map I_1 . Because of this action, the algorithm is called **Edge Directed Image Segmentation Algorithm.**

Algorithm 5.1 Edge Directed Image Segmentation

```
/*********************** Building the edges stack ***********************/
for each pixel in the edge map $E_1$ do
    if the pixel is an edge
        push the coordinates(x,y) into stack $Q_1$
    endif
endfor
/********************** The region merging process *********************/
while stack $Q_1$ is not empty do
    pop a pair of coordinates (x,y) in $Q_1$
    mark the position (x,y) in the image map $I_1$ to be visited
    make $g_1$ = the gray level at position (x,y) of $I_1$
    for each neighbor pixel $(x_1,y_1)$ of (x,y) in $I_1$
        make $g_2$ = its gray level of $(x_1,y_1)$ in $I_1$
        if not-visited and $abs(g_1 - g_2) < d$
            make $I_1(x_1,y_1) = g_1$
            make $I_1(x_1,y_1)$ to be visited
            endif
            if $E_1(x,y)$ is an edge
                  if $(abs(g_1 - g_2) < d)$ and not(abs(($I_1(x_1,y_1)$-$I_1(x,y)$))) max
                        push $(x_1,y_1)$ into stack $Q_1$
                  endif
            else
                        push $(x_1,y_1)$ into stack $Q_1$
             endif
        endfor
  enddo
```

Starting from the stack Q_1, merging of a homogeneous region begins. Each time an order pair (x_1,y_1) is pop from the stack Q_1, the gray levels in the image map I_1 is recorded as g_1 and the position are marked as *visited.* The next step of the algorithm is to check the neighbor pixels of (x_1,y_1) in I_1, if any of the neighbor pixels is *not visited* and the difference of the gray level with (x_1,y_1) is smaller than a preset value **d**, then the coordinates (x_1,y_1) are pushed into the stack Q_1 and the position (x_1,y_1) of I_1 are made equal to g_1. These processes are continued until the stack is empty.

Figure 1a The original 256 gray scale image

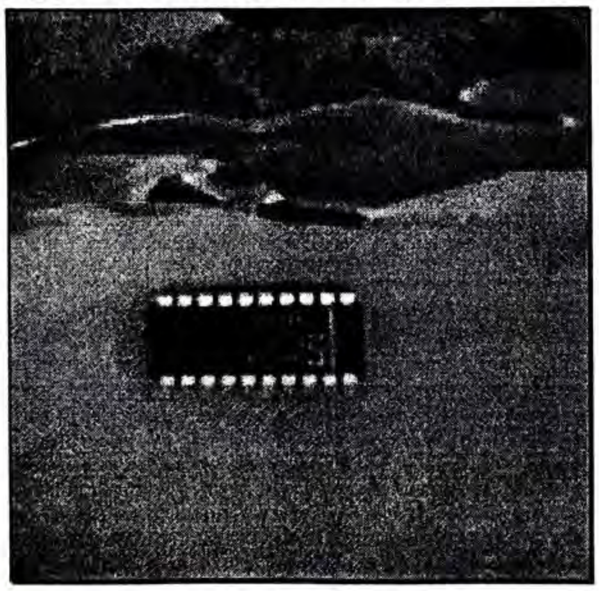


Figure 1b The input edge of the image 1a

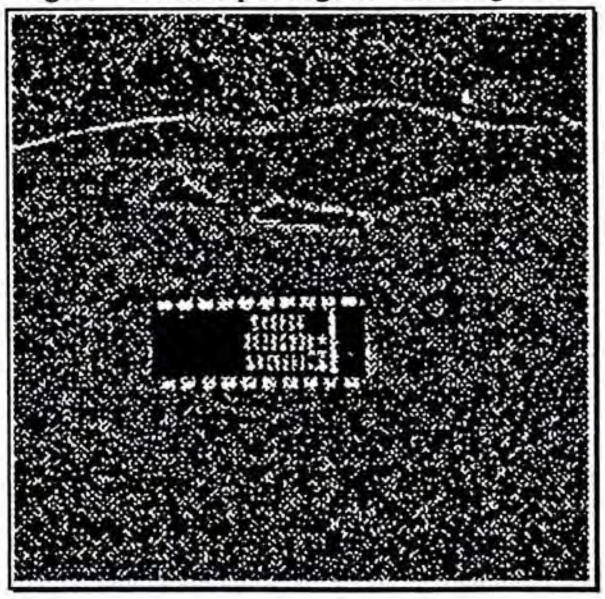


Figure 1c The segmented image with d = 50

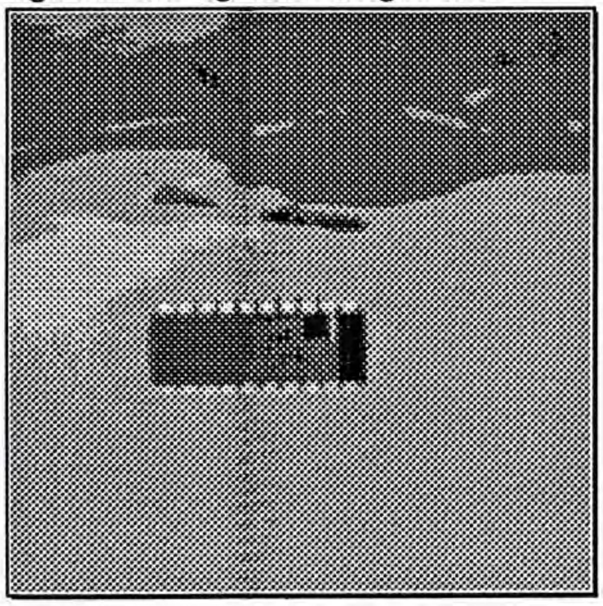


Figure 1d The segmented image with d = 70

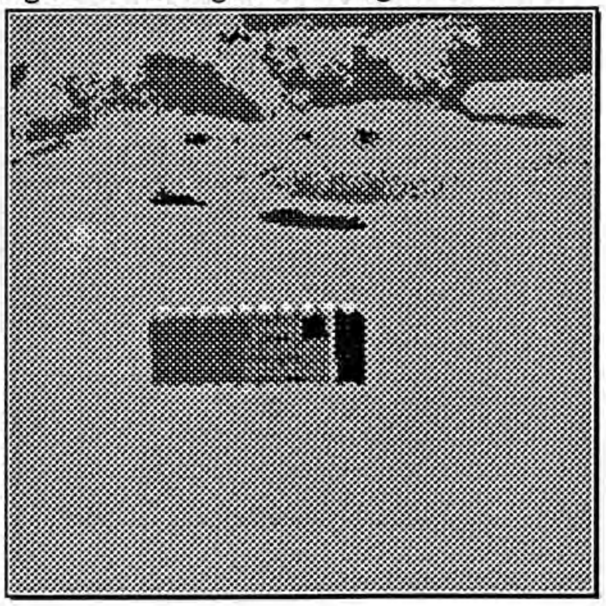


Figure 1a shows the original 256 gray scale image of size 256x256, Figure 1b is the edge obtained from the image processing architecture [Ref. 1]. Figure 1c is the image map obtained by applying Algorithm 1 with d = 50 and Figure 1d is the segmented image map with the same input images but different value of d = 70. Figure 2a to 2d are another set of images with the application of the same algorithm.

Figure 2a The original 256 gray scale image

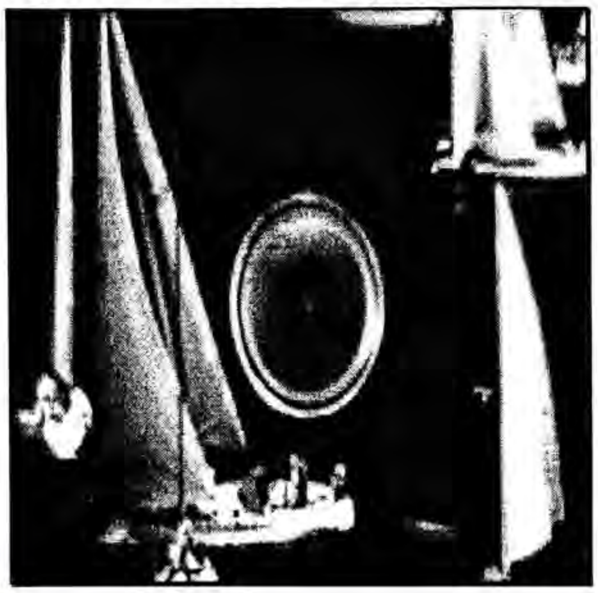


Figure 2b The input edge of the image 2a

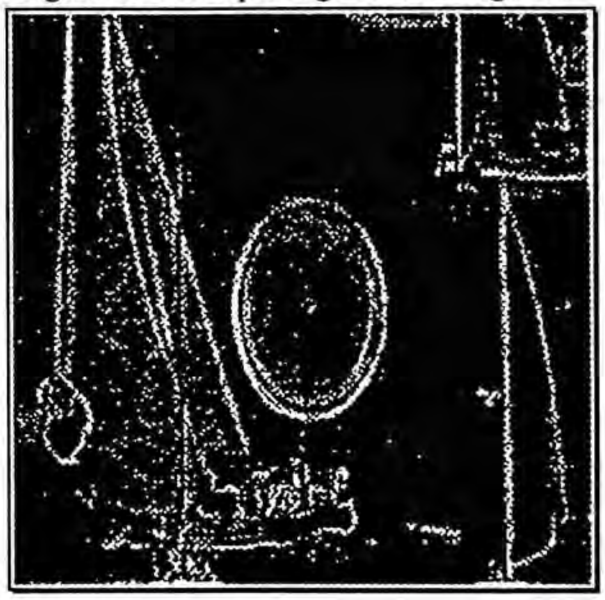


Figure 2c The segmented image with d = 80

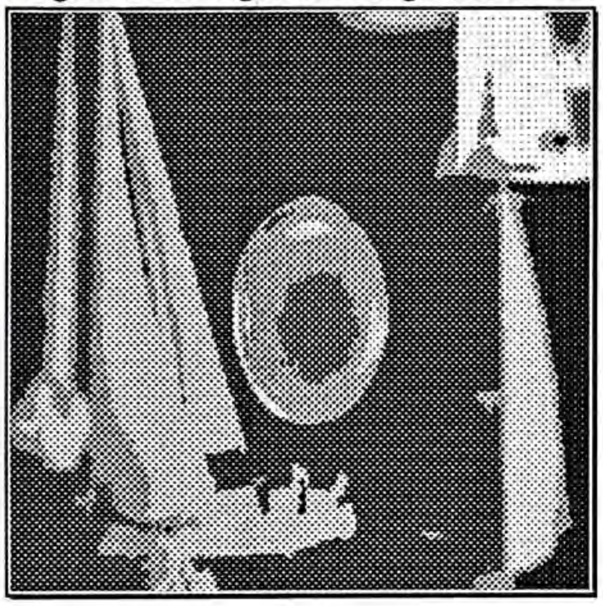


Figure 2d The segmented image with d=120

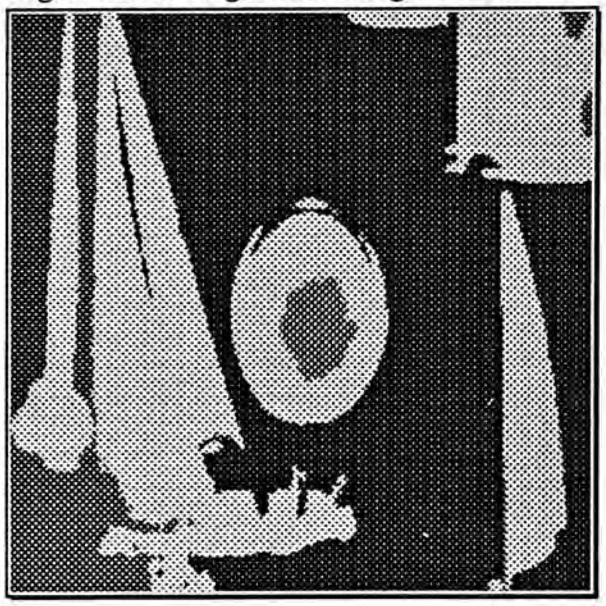


Figure 2a is the original 256 gray scale image of size 256x256, Figure 2b is the edge map of Figure 2a. Figure 2c is the segmented image map of Figure 2a by applying **Algorithm 4** with d = 80, Figure 2d is the segmented image with different d = 120. From the above examples, we observe that the larger the value d, the smaller of the number of regions can be obtained. This is the general trend, but not the unique phenomenon. There are some cases that with a small increment of d, the number of regions increases slightly. This situation is due to the uneven distribution of the gray level information in the original image map and the processing priority in popping the stack in Q_1 has been slightly altered, thus the order of the merging of the regions has been changed.

The main advantages of this algorithm are simple, no floating point calculation, each pixel is visited for only one time, there is no need to decide the merging criteria and finally it is totally localized, no need to process in higher level.

CHAPTER 6

FEATURE MATCHING ENHANCEMENT AND BUILDING OBJECT CORRESPONDENCES

6.1. Introduction

In the previous chapters, we have discussed that the transformation from bit map image to edges and then from edges to features. [Ref. 1]. The first step for the detection of motion is to build the feature correspondences between the image from time = t_n and time = t_{n+1}. It is called **Feature Matching**. This is accomplished by a simple iterative equation which will be discussed in next section. Even though the features are matched, we can only conclude that the feature correspondences are built, no information about the motion of a *whole object* can be deduced automatically by a machine (of course, the human brain can do so).

Although the rate of error from the feature matching is highly reduced due to the matching of features which is carefully selected from a feature classification matrix (or I will name it as a classification filter), there are still chances of mismatch. Since one object may have only one possibility of motion quantity, the true motion quantity (we say motion quantity we mean that the quantity of motion in horizontal [x] direction and the quantity of motion in vertical [y] direction, that is the order pair [x,y].) can be deduced by the voting of the concentration of the matched edge features. This type of voting is called **Subset Elimination** which is used as a tool for matching result enhancement.

The process of identification of individual objects involves the merging of similar gray level bit map images into homogeneous regions. This technique is called **Image Segmentation**. The homogeneous regions are treated as the projection of different **objets.** In this paper, a completely new algorithm named as **Edge Directed Image Segmentation** will be presented and discussed.

As mentioned above, there are errors due to mismatched features which can exist in **Layer 4.** However this problem can be solved by the technique of object voting which comes from image segmentation. The basic idea is that in one homogeneous region (we may consider it as an object), there is only one possibility of motion quantity. The solution of this problem is called **Object Conflict Resolution** which is also a new idea.

6.2 Attribute Matching from Feature Classification

With reference to the previous term paper [Ref. 1], once the edge features are classified and recorded in layer 3, feature matching can be carried out in layer 4. Layer 4 is a four dimensional layer. For the detection of translation in both horizontal (x), vertical (y) and the combination of the both, there are sub-layers in layer 4. Sub-layer [dx,dy] in layer 4 is responsible for the detection of translation of unit ***dx*** in horizontal and unit ***dy*** in vertical direction or their combination. The values of Layer 4 are governed by EQ(1)

$$L_4[dx,dy][i,j] = f_{43a}\,(f_{43b}(L_3[i,j],R_3[dx+i,dy+j])) \;\ldots\ldots\; \text{EQ(6.1)}.$$

There are 2 functions which are still not defined in EQ(1). Function f_{43b} will take 2 parameters as the input, the main purpose is to measure the similarity of the edge patterns. Since the edge patterns are transformed as the corresponding bit patterns [Ref. 1], one way to examine the similarity is to check the number of bits which are different. So, function f_{43b} can be defined as

$$f_{43b}\,(a,b) = a \text{ xor } b \text{ ...... (EQ 6.2)}$$

Where a and b is a member of the feature set *F*;

xor is the *exclusive OR* operator.

With this definition, the number of bits which **a** and **b** are different can be returned. But the difference of the number of bits of two edge attributes is the indication of the number of edges which are different. The number of bits of 1 returned by function f_{43b} indicates that there is a difference of 1 bit in the 2 input feature numbers **a** and **b**.

Since there may be noises existing in the input images, the features classified by layer3 may also be noisy, therefore inexact matching technique is needed to filter the noisy classified features. Function f_{43a} acts as a measurer for the degree of similarity of the 2 input features. It is defined as

$$f_{43a}\,(x) = 255 - 128 * n \quad \text{if } 0 <= n <= 2 \text{ .... (EQ 6.3)}$$

$$f_{43a}\,(x) = 0 \quad \text{for all other values of n}$$

Where n is the number of bit(s) set in x

According to this definition, a value of 255 returned by the function f_{43a} have the maximum degree of matching, 127 come after and all other values are discarded.

After the processing by (EQ 6.1) for all of the sub-layers in layer4, each sub-layer contains the raw information of matching between the image frames of time = $\mathbf{t_n}$ and time = $\mathbf{t_{n+1}}$. Now a clever researcher may raise the questions which have been introduced in the abstract of this paper. The solutions to these questions will be proposed in the following sections of this chapter.

6.3 Noise Reduction Using Subset Elimination

Even though the features are carefully selected from the edge map which is transformed from the image bit map, and the matching is carried out in the feature level, there are still chances for mismatch due to the massive number of features in one real world picture. In order to filter out noises, the technique called **Subset Elimination** has been applied to improve the matching results [Ref. 6.2] [Ref. 6.3].

Definition 6.1

Let $\mathbf{I}_1$ and $\mathbf{I}_2$ be the digitized feature maps of size **nxm**, **F** be a square filter of size **kxk**, where $k < n$ and $k < m$, then for the same position in $\mathbf{I}_1$ and $\mathbf{I}_2$, if the value returned by the convolution of the filter F in $\mathbf{I}_1$ is greater than in $\mathbf{I}_2$, we say that the region covered by the filter F in $\mathbf{I}_1$ is the **Superset of the same region** in $\mathbf{I}_2$ and the region covered by the filter F in $\mathbf{I}_2$ is the **Subset of the same region** in $\mathbf{I}_1$.

{End of Definition 6.1}

The basic philosophy for using **Subset Elimination** is based on the following observations:

1. For one object, only one quantity of motion is possible because the objects being considered are assumed to rigid bodies;

2. If an object has the motion quantity (x',y'), then there will be a cluster of feature matching results exist in the sub-layer (x',y') of layer4, all the matching results in other sub-layers are treated as noises;

3. The strength of matching of an object must be stronger than the strength of matching of the noises. This is reasonable, otherwise there will be no way to deduce the correctness of matching.

In practice, the filter F is selected such that when convolute will it, the result is sufficient to reflect the strength of matching of the region. Now let ***R*** be the region in a sub-layer in layer4 which is occupied by the filter F, the strength of matching in the region ***R*** is defined as :

$$S_f = \int_{x_1}^{x_2} \int_{y_1}^{y_2} (f_{p,q}) * L_4[x',y'](x,y)\, dx\, dy \text{ ...... (EQ 1.4a)}$$

Where the order pairs (x,y) are members of the region ***R***;

$f_{p,q}$ is the corresponding element of the filter *F*;

$L_4[x',y']$ is the [x',y']th sublayer of layer4.

When we consider digital images, (EQ 4a) can be written as :

$$S_f[x',y'][i,j] = \sum_{m=i-1}^{i+1} \sum_{n=j-1}^{j+1} (f_{m-i+2,n-j+2}) * L_4[x',y'][m,n] \text{ ...(EQ 6.4b)}$$

Where the order pairs [m,n] are members of the region ***R***;

$f_{m-i+2,n-j+2}$ is the corresponding element of the filter *F*;

$L_4[m,n]$ is the [m,n]th sublayer of layer4.

If we take the filter ***F*** to be

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{pmatrix}$$

The convolution result of $\boldsymbol{F}$ with a region in sublayers of layer4 is actually the summation of the raw matching results in that sublayer. Each value of S_f is the indication of the strength of matching in that cell in layer4.

For the same position in each sub-layers of layer4, only one cell can claim that the matching is correct, hence the strength indication S_f will act as a measure of voting for the cells in the corresponding position of each sublayers in layer4. By the **Philosophy of Subset Elimination**, only the cell with largest S_f can claim that it is successful to match the features there. The computation of S_f and the removal of mismatch are collectively called **Subset Elimination.**

6.4 Object Image Region Visiting Algorithm

Once the a gray level image map has been segmented into the corresponding segmented gray level image map, the identification of the objects in the segmented image needs the understanding of the regions in the segmented image. The understanding of the regions in the segmented image can be accomplished by visiting all the member pixels of a region in the segmented image map.

In this section, a region visiting algorithm is presented. The algorithm visits each pixel in the segmented image, when there is a change of gray level, it realizes the fact that there is a change of region. If there is a change of region, the coordinates of that pixel are added into a list. When the process is finished, the algorithm returns the number of regions and a list of which its elements contain a pair of coordinates which are the positions of the corresponding region (Since a region has homogeneous gray level, one position is enough to trace the whole region).

This algorithm takes the input of the segmented image S_1, initializes the list L_1 to be empty. Starting from the first row, first column of the segmented image, add the coordinates of the first pixel to the list, mark the pixel to be *visited.* The algorithm then visits each of the surrounding pixels of that pixel, if the *gray level is the same* and the pixel is *not visited*, the coordinates of the pixel are pushed into a stack Q_1. Initially, all the pixels in the segmented image are marked *not visited.* After visiting the first pixel, the algorithm pops out an element in the stack Q_1, for each of the surrounding pixels of that pixel, if the *gray level is the same* and the pixel is *not visited* the coordinates of the pixel are pushed into a stack Q_1. This process continues until all the pixels of the region are visited. The algorithm then goes to the region, add the coordinates of the this region to the list L_1. The region visiting process continues until all the pixels of this region are visited. The algorithm continues until all the pixels in the segmented images are visited.

Algorithm 6.1 Object Image Region Visiting Algorithm

```
/**************** Object Image Region Visiting Algorithm ***************/
$g_1 = S_1[0,0]$
n = 0
for $i$ = 0 to *maxrow*
    for $j$ = 0 to *maxcol*
        if $S_1[i,j]$ is *not-visited* and $S_1[i,j]$ <> $g_1$
            mark $S_1[i,j]$ to be *visited*
            push [i,j] into stack $Q_1$
            add [i,j] to the list $L_1$
```

```
                n = n + 1
                $g_1$ = $S_1$[i,j]
                while stack $Q_1$ is not empty do
                    pop one element in $Q_1$ as (x,y)
                    for each surrounding pixel of (x,y) do
                        if $S_1$[i,j] is *not-visited* and $S_1$[i,j] = $g_1$
                            mark $S_1$[i,j] to be *visited*
                            push [i,j] into stack $Q_1$
                        endif
                    endfor
                endwhile
            endif
        endfor
    endfor
```

Where *maxrow* and *maxcol* are the number of rows and columns of the segmented image respectively;

L_1 is the list of coordinates of the first element of a region;

n is the number of regions returned by the algorithm;

6.5 Object Conflict Resolution and Translation Detection

As mentioned before, for the raw matching results in the sub-layers of layer4, there are noises of mismatching information, the elimination of the mismatching information needs the identification of the whole object.

In the last section, the original images are segmented into a number of homogeneous regions, each homogeneous region can be treated as the image of an object. Since each object can have only one motion quantity, then for all the sub-layers in layer4, only one correct matching can exist in one sublayer of layer4 for every region. Based on this rationale, we can formulate the problem as shown below :

Let S be the segmented image of size $m \times n$;

$L_4[dx,dy]$ be a sublayer of layer4 of size $m \times n$ which is responsible for the detection of objects of translation quantity $[dx,dy]$;

R_i be the ith region of the segmented image S;

The order pair of coordinates $[x_i,y_i]$ is a member of R_i;

$L_4[dx,dy][i,j]$ is a processing unit at the position $[i,j]$ of $L_4[dx,dy]$.

Definition 6.2

A matching of region R_i is said to be *in conflict* if it occurs at $[x_i,y_i]$ of $L_4[dx,dy]$ and occurs at $[x'_i,y'_i]$ of $L_4[dx',dy']$.

where $[x_i,y_i] \in R_i$ and $[x'_i,y'_i] \in R_i$ and NOT $((x_i = x'_i)$ and $(y_i = y'_i))$.

{End of definition 6.2}

Observation 6.1

For any R_i of the segmented image S, if matching occurs at $[x_i,y_i] \in R_i$ of $L_4[dx,dy]$ and matching occurs at $[x'_i,y'_i] \in R_i$ of $L_4[dx',dy']$, then the order pairs $[dx,dy]$ and $[dx',dy']$ must be identical. That is $(dx = dx')$ and $(dy = dy')$.

{End of observation 6.1}

Observation 6.1 is called the unique motion quantity assumption of a rigid body. The important meaning of the observation is to emphasize the fact that when we take an object as a whole, if the motion quantity of one part of a rigid object is **[x,y]**, then the motion quantity of all other parts of the object must also be **[x,y]**.

If **observation 6.1** is not satisfied, then either they are of different objects or there are mismatched features occurring in the same region. Now the focus will be switched to that how to identify the objects in the original map. This problem has already been solved by the image segmentation algorithm described in the last section.

It is very common for an image matching algorithm to have mismatch. Although the process of feature classification in Layer3 [Ref. 1] have classified the edge features into feature numbers which has taken the spatial information into consideration, a very large portion of mismatch possibility has been eliminated, there are still chances of mismatch. **Object Conflict Resolution** is the final tool for the elimination of the mismatched objects.

The main idea for the method of **Object Conflict Resolution** is based on **Observation 1**. The followings are the formulations for **Object Conflict Resolution**.

Definition 6.3

$SM_i[dx,dy]$ is the strength of matching of a region R_i in sublayer **[dx,dy]** of layer4 is defined as :

$$SM_i[dx,dy] = \int_{x_1}^{x_2} \int_{y_1}^{y_2} dxMdy(x,y)\ dx\ dy \text{ ...... (EQ 6.5)}$$

For all (x,y) which are members of R_i.

Where **dxMdy(x,y)** is matching result at position (x,y) of sublayer [dx,dy] of layer4 which is returned by **Algorithm 1.**

{End of definition 6.3}

Definition 6.4

$SM_i[dx,dy]$ is said to be *stronger* than $SM_i[dx',dy']$ if $SM_i[dx,dy] > SM_i[dx',dy']$. {End of definition 6.4}

Algorithm 6.2 **Object Conflict Resolution and Translation Detection**

```
/********** Object Conflict Resolution and Translation Detection ***********/
maxM = 0
for each region $R_i$ of the segmented image do
    for i = 0 to maxdx
        for j = 0 to maxdy
            visit all $(x,y) \in R_i$ in $L_4[i,j]$
            calculate $SM_i[i,j]$
        endforj
    endfori
    if $SM_i[i,j] > maxM$
        $maxM = SM_i[i,j]$
        visit all the pixels of $R_i$ in layer4 $L_4[i,j]$
        mark the value of $L_4[i,j][x,y]$ to HIGH for all $(x,y) \in R_i$
    endif
endfor
```

Algorithm 6.2 is clear and self-explaining. After the application of **Algorithm 6.2** in the sub-layers of layer4, all the translation situations of all objects which can be identified in the segmented image are detected. If an object has the motion quantity **[dx,dy]**, then the shape of the object can be identified in the sublayer[dx,dy] of layer4. By the shape of the object, we mean the homogeneous region which can be identified in the segmented image. The experimental results and the discussion of the topic involving the detection of rigid body translation will be given in next chapter.

CHAPTER 7

EXPERIMENTAL RESULTS DISCUSSION, AND CONCLUSION

In this chapter, the experimental results are presented in Section 7.1 and 7.2., the analysis of accuracy for the proposed model by using the real pictures is also given in Section 7.2. In Section 7.3, there will be the qualitative comparisons with the methods published in the current related literature. Finally, the discussion will be given.

7.1 Two Sets of Experimental Results

The figures below are two sets of experimental results. The results are quite satisfactory (there is no comparable result can be found in the literature). **Figure 7.1a** shows an image map which contain some objects, this image is taken at **time = $\mathbf{t_n}$**. Figure 7.2a shows another image map which are taken in the same scenery at **time = $\mathbf{t_{n+1}}$** (where $t_{n+1} = t_n + \delta t$). There are some movements for each of the objects. The images are then passed to the architecture for translation detection. The results are obtained as shown by Figure 7.3a to 7.3e. The motion quantities [1,0] and [3,3] are detected by Figure 7.3c and Figure 7.3d respectively.

Figure 7.1b is the edge of Figure 7.1a which is obtained by the processing of the connection between Layer 1 and Layer 2. Figure 7.1c is the features (Shown by intensity) obtained in layer 3. Figure 7.3b to Figure 7.3e is the final results obtained in Layer 4. ***White color means motion is detected in that sublayer, black color means no motion is detected.***

In Figure 7.3c, a boat is detected with motion quantity [1,0] (i.e. x=1, y=0), a circular object is detected with motion quantity [3,3] (i.e. x=3, y=3). For the sublayer [0,0] of layer4, the white color indicates the background, no motion is detected. Black color means that there are some objects whose motion quantity is not [0,0].

Figure 7.1.1a Image at time = $\mathbf{t_n}$

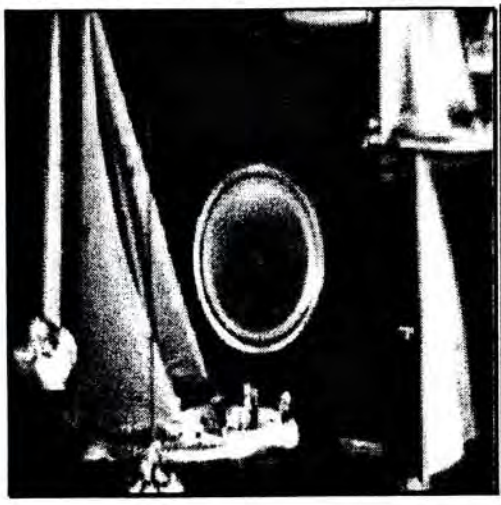


Figure 7.1.1b Edge at time = $\mathbf{t_n}$

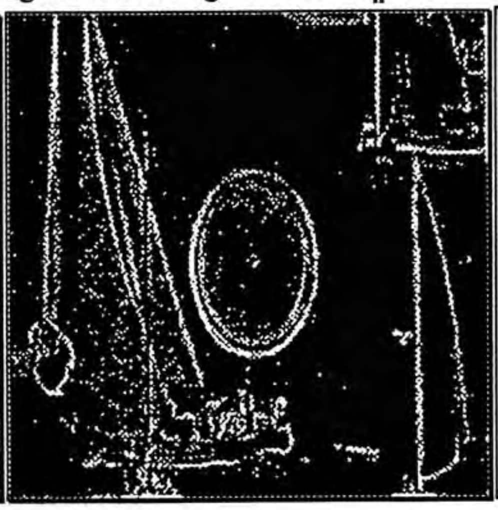


Figure 7.1.1c Features at time = $\mathbf{t_n}$

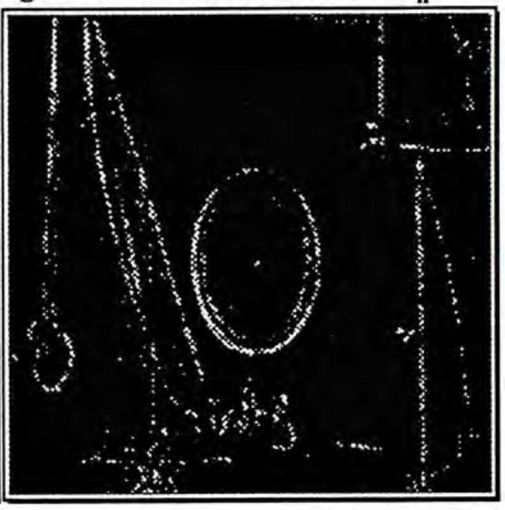


Figure 7.1.2a Image at time = $\mathbf{t_{n+1}}$

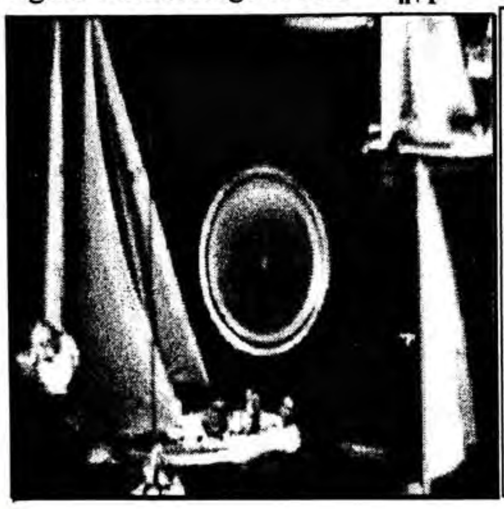


Figure 7.1.2b Edge at time = $\mathbf{t_{n+1}}$

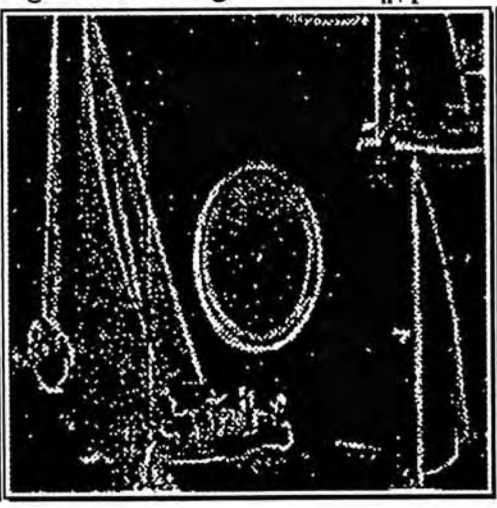


Figure 7.1.3c Feature at time = $\mathbf{t_{n+1}}$

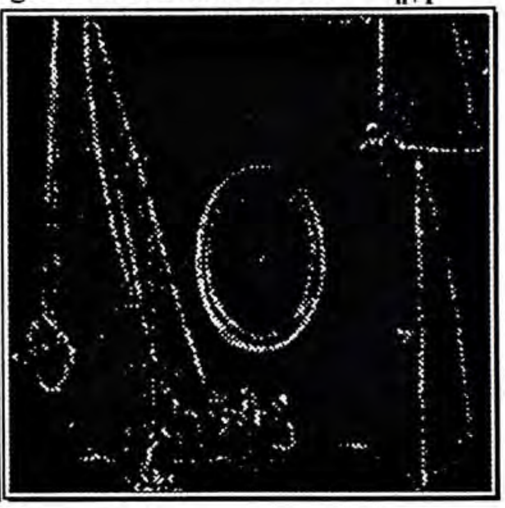


Figure 7.1.3a Segmented image of 7.1.1a

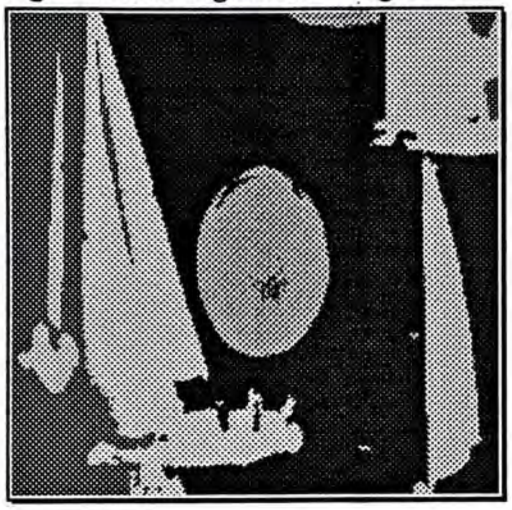


Figure 7.1.3b Sublayer $L_4[0,0]$

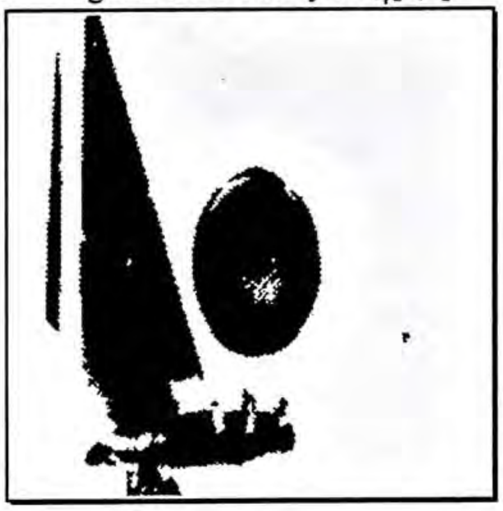


Figure 7.1.3c Sublayer $L_4[1,0]$

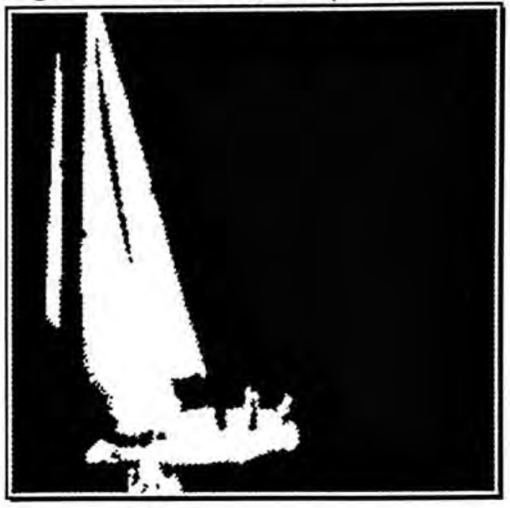


Figure 7.1.3d Sublayer $L_4[3,1]$

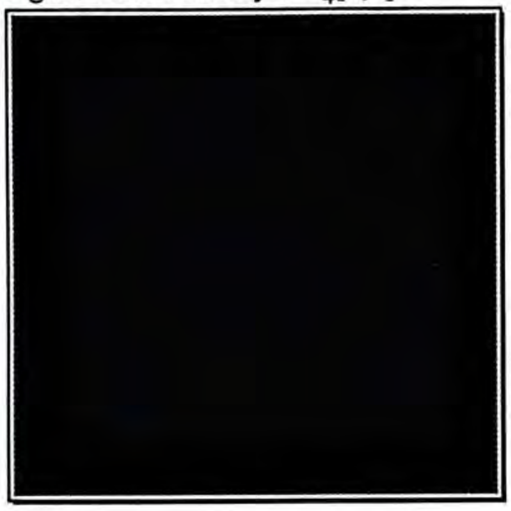


Figure 7.1.3d Sublayer $L_4[3,3]$

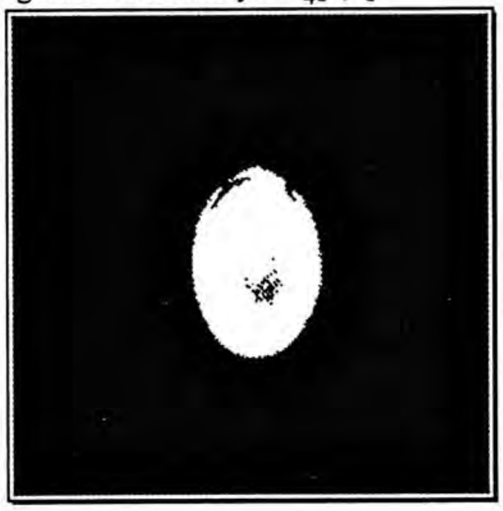


Figure 7.1.3e All other Sublayers

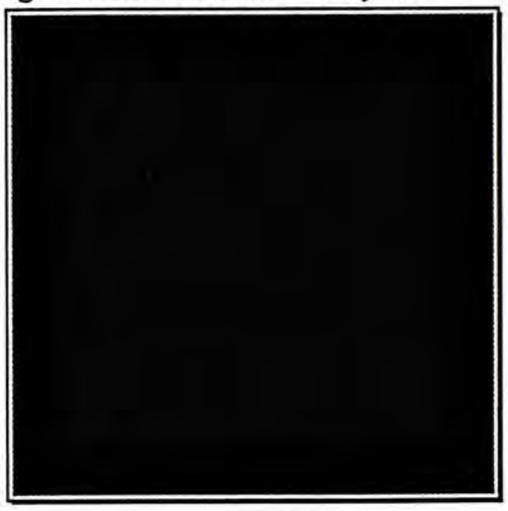


Figure 7.1.4a Image at time = $\mathbf{t_n}$

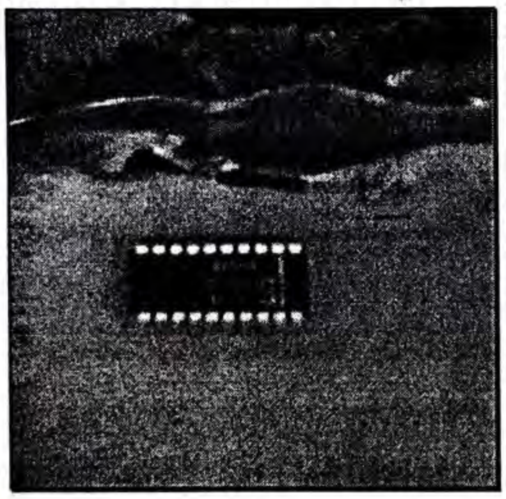


Figure 7.1.4b Edge at time = $\mathbf{t_n}$

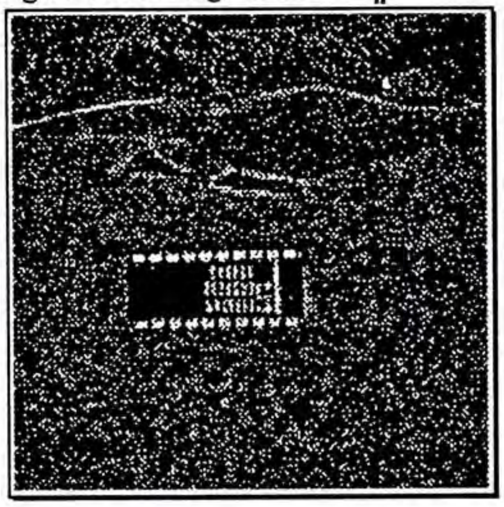
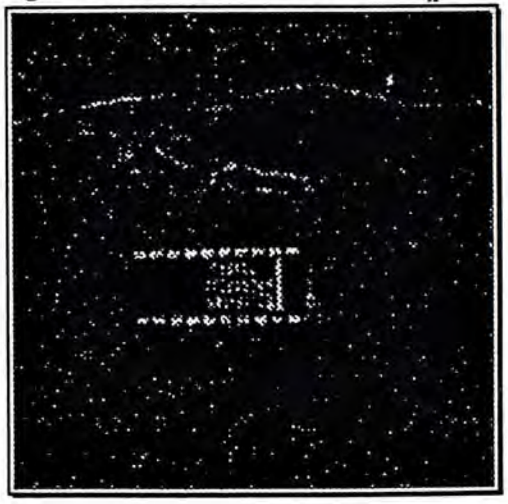


Figure 7.1.4C Features at time = $\mathbf{t_n}$



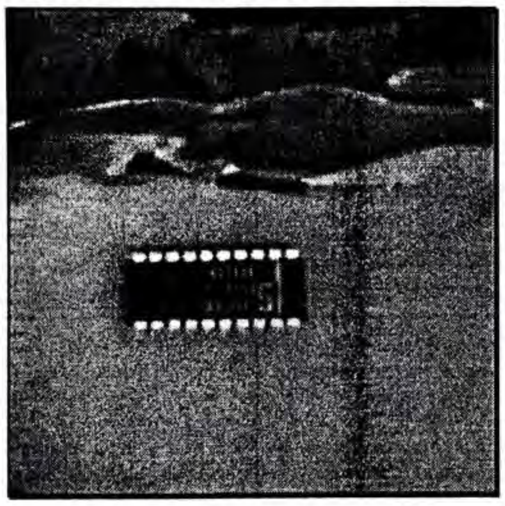


Figure 7.1.5a Image at time = t_{n+1}

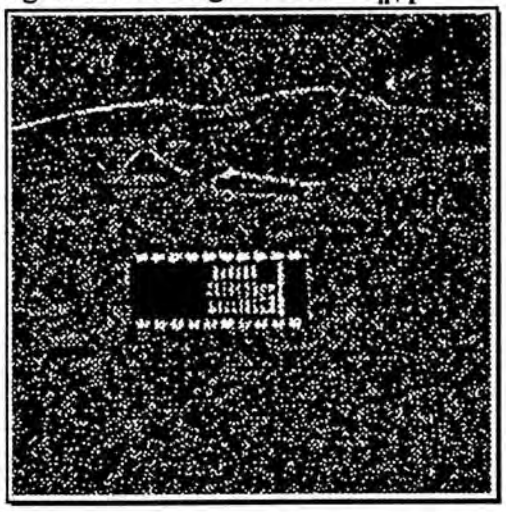


Figure 7.1.5b Edge at time = t_{n+1}

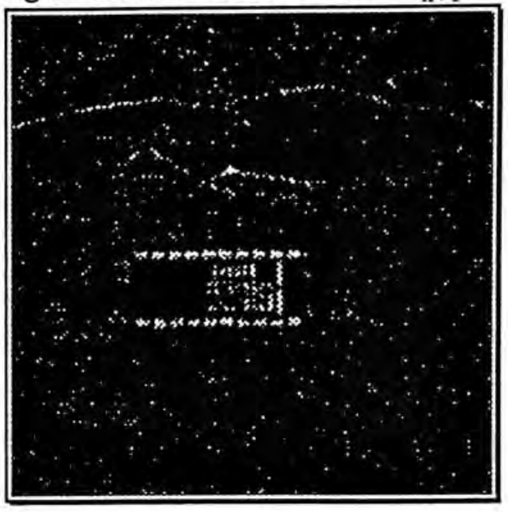


Figure 7.1.5c Features at time = t_{n+1}

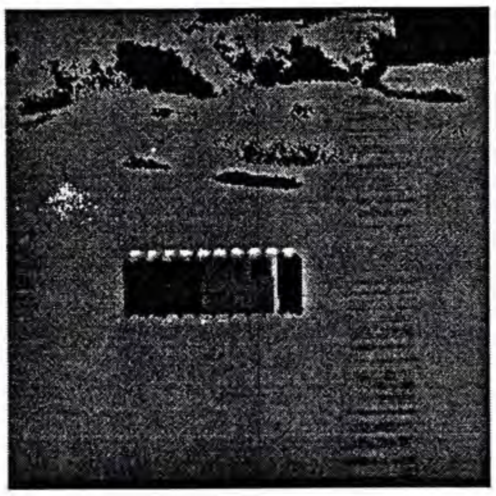


Figure 7.1.6a Segmented image of 7.1.4a

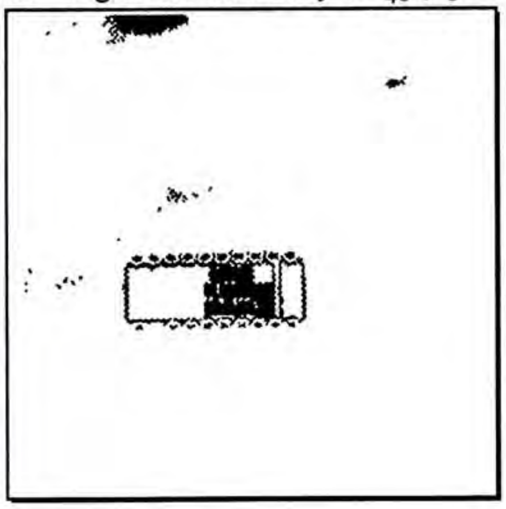


Figure 7.1.6b Sublayer $L_4[0,0]$

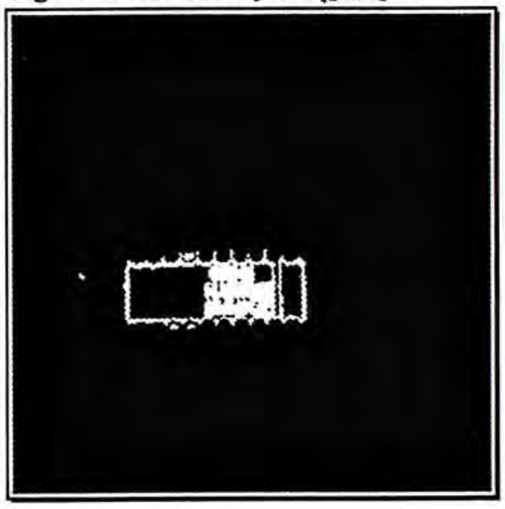


Figure 7.1.6c Sublayer $L_4[2,1]$

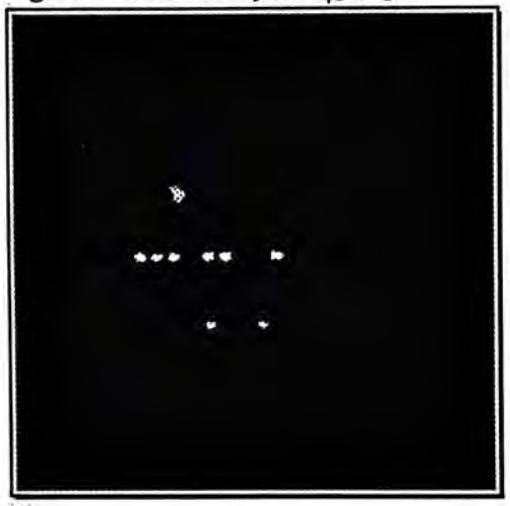


Figure 7.1.6d Sublayer $L_4[2,2]$

Figure 7.1.6e Sublayer $L_4[2,3]$

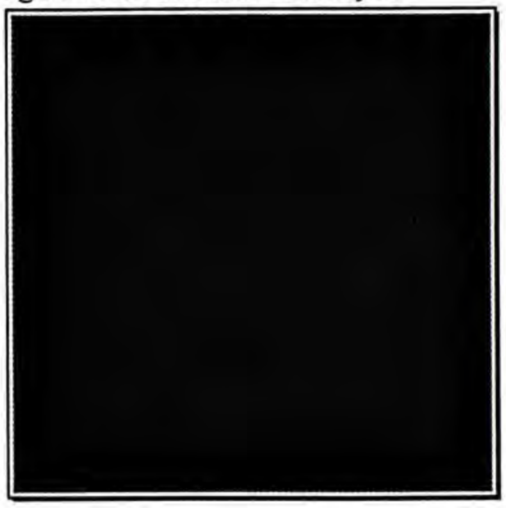


Figure 7.1.6f All other Sublayers

Figure 7.4a to Figure 7.4c and Figure 7.5a to Figure 7.5c are another two sets of images which are taken under the same situations. Figure 7.4a is taken at **time = t_n**, Figure 7.5a is taken at **time = t_{n+1}** . The two sets of images are passed to the architecture for translation detection. The results are obtained as shown by Figure 7.6a to Figure 7.6f. The interpretation is the same as before.

7.2 Accuracy Evaluation of the Proposed Method

In order to evaluate the accuracy and consistency of our proposed methods, two sets of consecutive pictures (totally 16 pictures) are used for testing. The first set of consecutive pictures is as shown below from Figure 7.2.1a to Figure 7.2.1b where a circular object is performing translation motion. The motion detection is being carried out by using every two consecutive pictures.

The figures have been taken in such a manner that Figure 7.2.1a has been taken at time = t_1, Figure 7.2.1b has been taken at time = t_2, Figure 7.2.1c has been taken at time = t_3, Figure 7.2.1d has been taken at time = t_4, Figure 7.2.1e has been taken at time = t_5, Figure 7.2.1f has been taken at time = t_6, Figure 7.2.1g has been taken at time = t_7 Figure 7.2.1h has been taken at time = t_8 (where $t_{n+1} = t_n + \delta t$).The actual translation quantity (in pixel) has been obtained by using a program to overlap the object region and then judged by human eyes. The accuracy results obtained by the proposed model and the actual results are tabulated as Table 7.2a. The resulting sublayers in layer 4 are as shown by Figure 7.2.2a to Figure 7.2.2g.

Table 7.2a Accuracy of the proposed model against human eyes.

	Fig. 7.2.1b	**Fig. 7.2.1c**	**Fig. 7.2.1d**	**Fig. 7.2.1e**	**Fig. 7.2.1f**	**Fig. 7.2.1g**	**Fig. 7.2.1h**
Fig. 7.2.1a (Detected)	[2,0] **(Detected)**	[2,1] **(Detected)**	[3,3] **(Detected)**	[3,4] **(Detected)**	[4,4] **(Detected)**	[5,4] **(Detected)**	[5,5] **(Detected)**
Fig. 7.2.1a (Actual)	[2,0] **(Actual)**	[2,1] **(Actual)**	[3,3] **(Actual)**	[3,4] **(Actual)**	[5,4] **(Actual)**	[5,4] **(Actual)**	[6,6] **(Actual)**
Error	[0,0]	[0,0]	[0,0]	[0,0]	[1,0]	[0,0]	[1,1]

From the results obtained in Table 7.2a, the performance of our proposed model is acceptable and is usually consistent with the observation of human eyes. However, an error of value ±1 for x-direction, y-direction or both may occur. This is due to the difference of the decision criteria of our proposed method and the decision criteria of the human eyes, or sometimes it may be the hardware equipment error occurred at any where in the detection procedures starting from taking the photoes to the processing results obtained.

Figure 7.2.1a Time = t_1

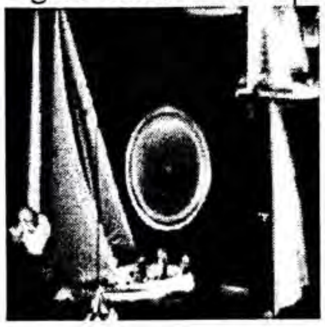


Figure 7.2.1b Time = t_2

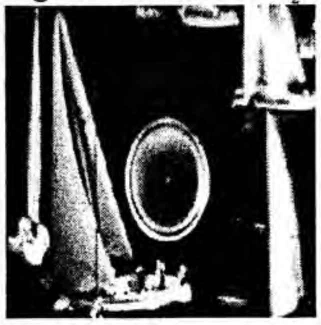


Figure 7.2.1c Time = t_3

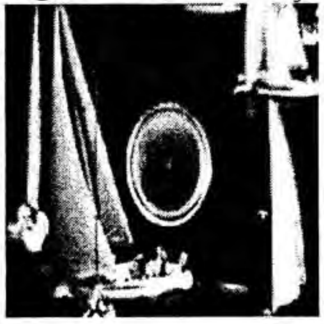


Figure 7.2.1d Time = t_4

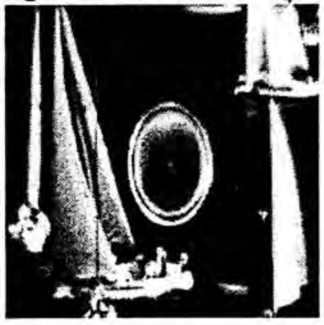


Figure 7.2.1e Time = t_5

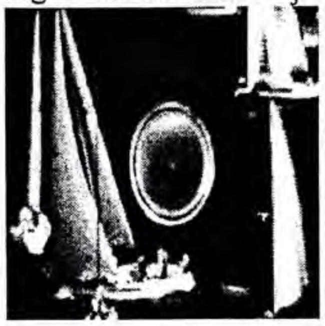


Figure 7.2.1f Time = t_6

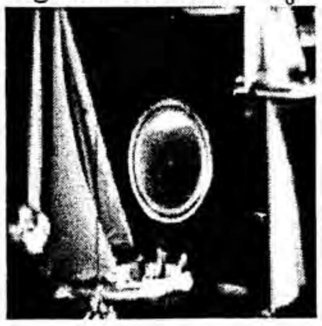


Figure 7.2.1g Time = t_7

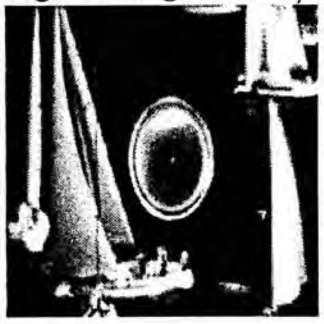
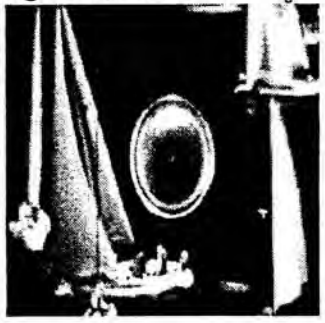


Figure 7.2.1h Time = t_8



The Results Detected by the Proposed model. (Only the interested Sublayers are shown)

Figure 7.2.2a
7.2.1a with 7.2.1b
Sublayer $L_4[2,0]$

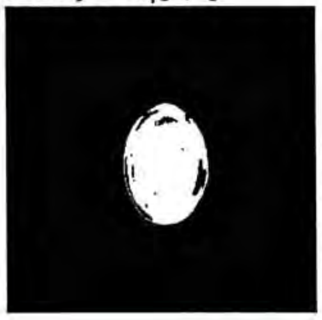


Figure 7.2.2b
7.2.1a with 7.2.1c
Sublayer $L_4[2,1]$

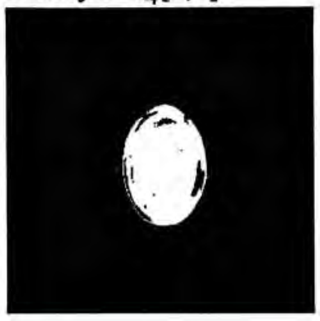


Figure 7.2.2c
7.2.1a with 7.2.1d
Sublayer $L_4[3,3]$

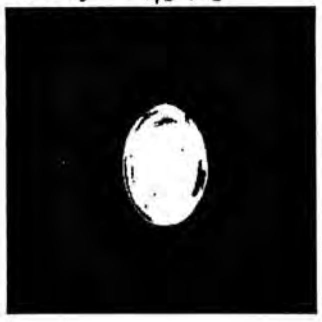


Figure 7.2.2d
7.2.1a with 7.2.1e
Sublayer $L_4[3,4]$

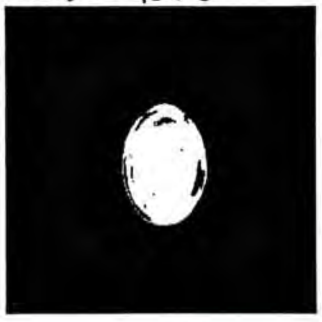


Figure 7.2.2e
7.2.1a with 7.2.1f
Sublayer $L_4[4,4]$

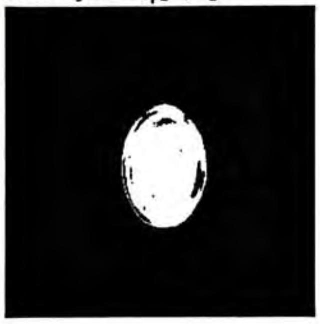


Figure 7.2.2f
7.2.1a with 7.2.1g
Sublayer $L_4[5,4]$

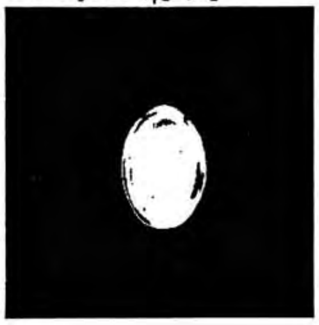


Figure 7.2.2g
7.2.1a with 7.2.1h
Sublayer $L_4[5,5]$

Figure 7.2.3a
7.2.1b with 7.2.1c
Sublayer $L_4[1,1]$

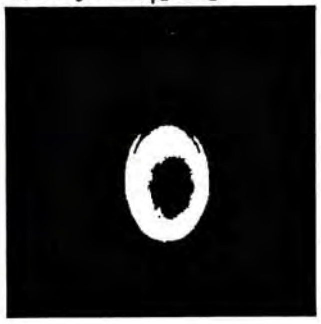


Figure 7.2.3b
7.2.1b with 7.2.1d
Sublayer $L_4[1,3]$

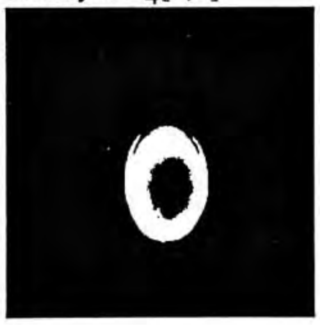


Figure 7.2.3c
7.2.1b with 7.2.1e
Sublayer $L_4[1,4]$

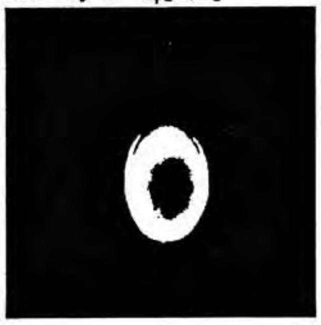


Figure 7.2.3d
7.2.1b with 7.2.1f
Sublayer $L_4[2,4]$

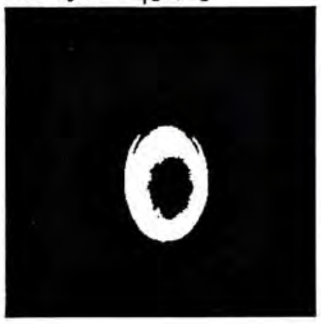


Figure 7.2.3e
7.2.1b with 7.2.1g
Sublayer $L_4[2,5]$

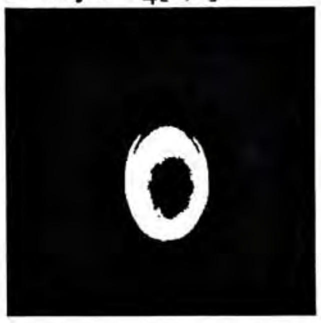


Figure 7.2.3f
7.2.1b with 7.2.1h
Sublayer $L_4[3,5]$

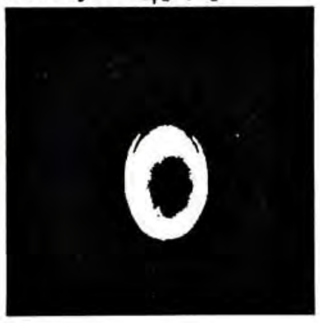


Figure 7.2.4a
7.2.1c with 7.2.1d
Sublayer $L_4[1,2]$

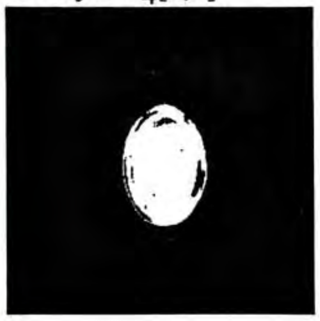


Figure 7.2.4b
7.2.1c with 7.2.1e
Sublayer $L_4[1,3]$

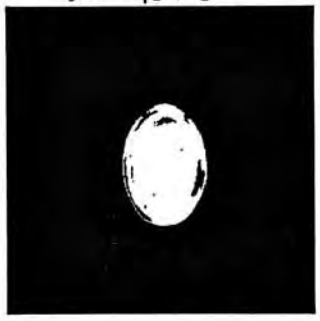


Figure 7.2.4c
7.2.1c with 7.2.1f
Sublayer $L_4[2,3]$

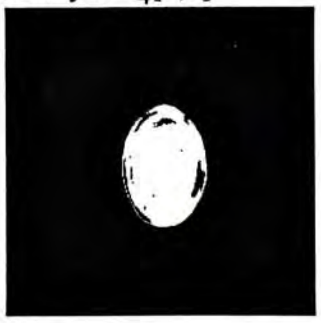


Figure 7.2.4d
7.2.1c with 7.2.1g
Sublayer $L_4[3,3]$

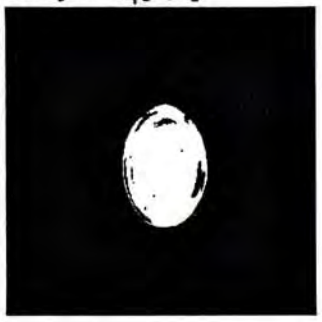


Figure 7.2.4e
7.2.1c with 7.2.1h
Sublayer $L_4[3,4]$

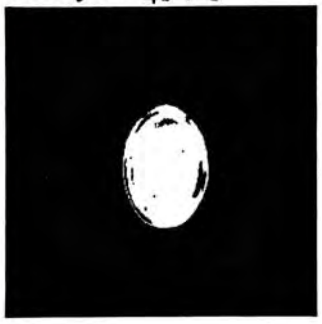


Figure 7.2.5a
7.2.1d with 7.2.1e
Sublayer $L_4[1,1]$

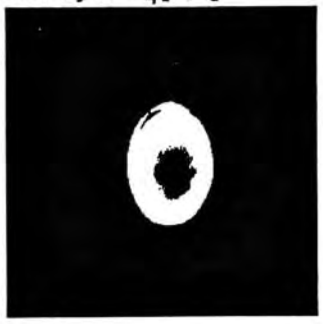


Figure 7.2.5b
7.2.1d with 7.2.1f
Sublayer $L_4[1,1]$

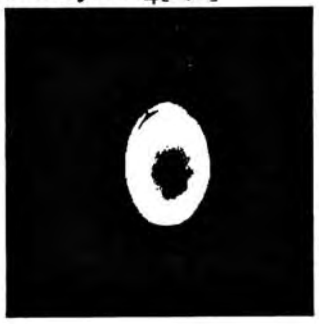


Figure 7.2.5c
7.2.1d with 7.2.1g
Sublayer $L_4[2,1]$

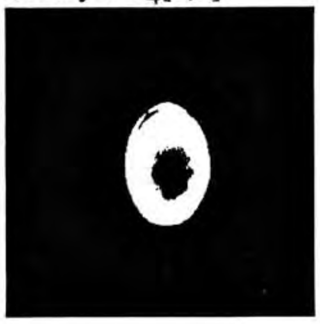


Figure 7.2.5d
7.2.1d with 7.2.1h
Sublayer $L_4[2,2]$

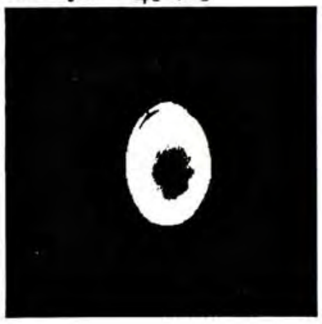


Figure 7.2.6a
7.2.1e with 7.2.1f
Sublayer $L_4[1,0]$

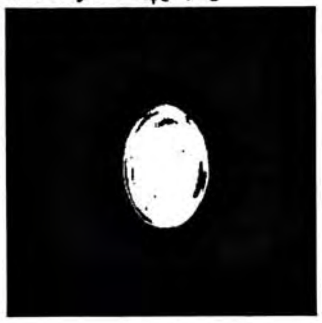


Figure 7.2.6b
7.2.1e with 7.2.1g
Sublayer $L_4[2,0]$

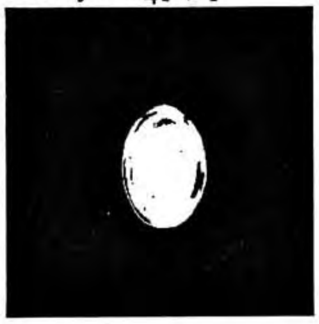


Figure 7.2.6c
7.2.1e with 7.2.1h
Sublayer $L_4[2,1]$

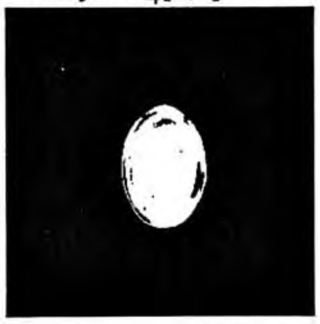


Figure 7.2.7a
7.2.1f with 7.2.1g
Sublayer $L_4[1,0]$

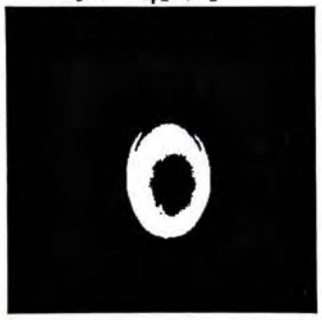


Figure 7.2.7b
7.2.1f with 7.2.1h
Sublayer $L_4[1,1]$

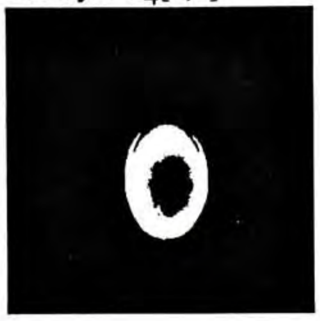
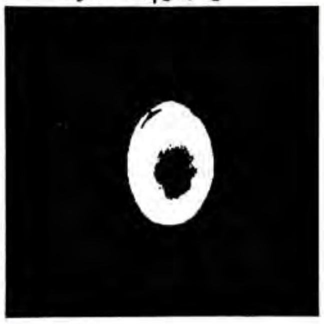


Figure 7.2.8a
7.2.1f with 7.2.1g

Sublayer $L_4[0,1]$



Form the results obtained from Figure 7.2.2a to Figure 7.2.8a, the error analysis of our proposed model can be carried out and the results are tabulated as Table 7.2b.

Table 7.2b Error checking across each picture from our proposed model.

	Fig. 7.1.1b	Fig. 7.1.1c	Fig. 7.1.1d	Fig. 7.1.1e	Fig. 7.1.1f	Fig. 7.1.1g	Fig. 7.1.1h
Fig. 7.1.1a	Error [0,0]	Error [0,0]	Error [0,0]	Error [0,0]	Error [1,0]	Error [0,0]	Error [1,1]
Fig. 7.1.1b		Error [1,0]	Error [0,0]	Error [0,0]	Error [0,0]	Error [1,1]	Error [0,0]
Fig. 7.1.1c			Error [0,0]	Error [0,0]	Error [0,0]	Error [0,0]	Error [0,0]
Fig. 7.1.1d				Error [1,0]	Error [0,0]	Error [0,0]	Error [0,0]
Fig. 7.1.1e					Error [0,0]	Error [0,0]	Error [0,0]
Fig. 7.1.1f						Error [0,0]	Error [0,0]
Fig. 7.1.1g							Error [0,0]

* Error [x,y] This is the indicator for the error, 0 value for x, y means no error, any other integer for x and y means the discrepancy is of x pixels in horizontal direction and y pixels in vertical direction.

The same model and processing procedures are applied to the following set of 8 consecutive pictures which are also taken at different consecutive time intervals. The images are taken in the same manner as Figure 7.2.1a to Figure 7.2.1h. The detected results are tabulated as Table 7.2c and Table 7.2d. All the procedures and the explanation of the results are as before.

Table 7.2c Accuracy of the proposed model against human eyes.

	Fig. 7.2.9b	**Fig. 7.2.9c**	**Fig. 7.2.9d**	**Fig. 7.2.9e**	**Fig. 7.2.9f**	**Fig. 7.2.9g**	**Fig. 7.2.9h**
Fig. 7.2.9a **(Detected)**	[2,2] **(Detected)**	[3,5] **(Detected)**	[5,4] **(Detected)**	[6,6] **(Detected)**	[7,7] **(Detected)**	[7,7] **(Detected)**	[8,9] **(Detected)**
Fig. 7.2.9a **(Actual)**	[2,2] **(Actual)**	[3,4] **(Actual)**	[5,4] **(Actual)**	[6,6] **(Actual)**	[7,6] **(Actual)**	[7,7] **(Actual)**	[8,9] **(Actual)**
Error	[0,0]	[0,1]	[0,0]	[1,0]	[0,0]	[0,0]	[0,0]

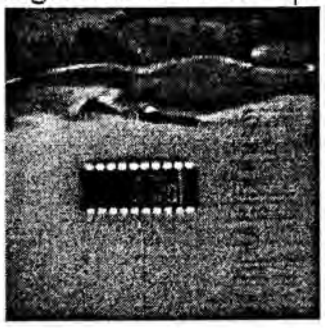
Figure 7.2.9a Time = t_1


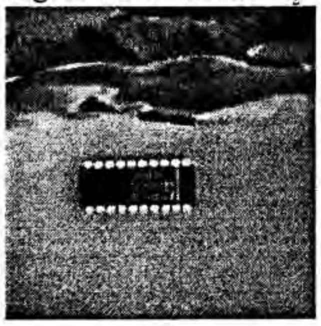
Figure 7.2.9b Time = t_2


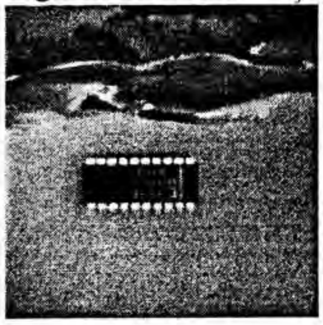
Figure 7.2.9c Time = t_3


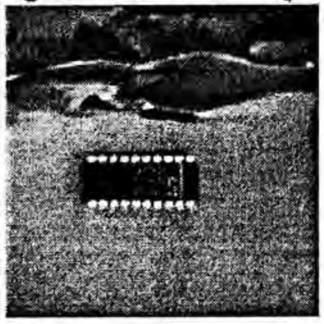
Figure 7.2.9d Time = t_4


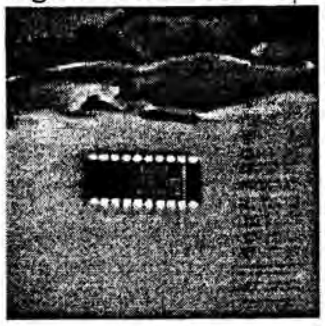
Figure 7.2.9e Time = t_1


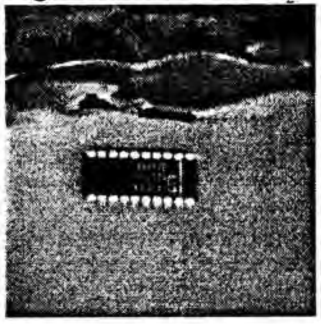
Figure 7.2.9f Time = t_2


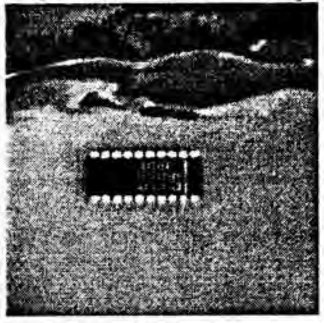
Figure 7.2.9g Time = t_3


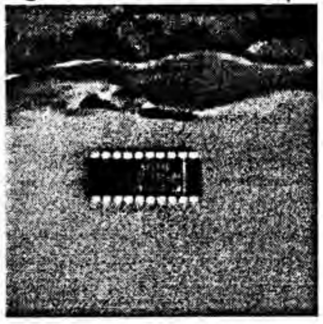
Figure 7.2.9h Time = t_4


Table 7.2d Error checking across each picture from our proposed model.

	Fig. 7.1.9b	Fig. 7.1.9c	Fig. 7.1.9d	Fig. 7.1.9e	Fig. 7.1.9f	Fig. 7.1.9g	Fig. 7.1.9h
Fig. 7.1.9a	Error [0,0]	Error [0,0]	Error [0,0]	Error [1,0]	Error [0,0]	Error [0,0]	Error [1,1]
Fig. 7.1.9b		Error [0,0]	Error [0,0]	Error [0,0]	Error [0,0]	Error [0,1]	Error [0,0]
Fig. 7.1.9c			Error [0,0]	Error [0,0]	Error [0,0]	Error [0,0]	Error [0,0]
Fig. 7.1.9d				Error [0,0]	Error [0,0]	Error [0,0]	Error [0,0]
Fig. 7.1.9e					Error [0,0]	Error [1,0]	Error [0,0]
Fig. 7.1.9f						Error [0,0]	Error [0,0]
Fig. 7.1.9g							Error [0,0]

* Error [x,y]	This is the indicator for the error, 0 value for x, y means no error, any other integer for x and y means the discrepancy is of x pixels in horizontal direction and y pixels in vertical direction.

From the experimental results and the error tables Table 7.2b and Table 7.2d, the error analysis of the proposed model can by estimated and tabulated as Table 7.2e.

Table 7.2e Error analysis of the experimental results.

	Counts	**Total Trials**	**Occurrence Percentage**
No Error	**48**	**56**	**85.7%**
Entries within one pixel of error	**8**	**56**	**14.3%**
Entries more than one pixel of error	**0**	**56**	**0%**

From Table 7.2e, we can see that the percentage for no error is 85.7%, the error for ±1 pixel is 14.3% and all other types of error is 0%. Although the accuracy results are satisfactory, a quantity of ±1 pixel for errors can still exist. The error quantity for ±1 pixel may due to the sensitiveity of the methods proposed or the difference in decision criteria for human eyes and the proposed algorithm.

In next section, the comparison of the proposed method against the methods which are published in the current related literatures will be given qualitatively as no quantitative data can be obtained. The comparison will mainly concentrate on the processing methods, the advantages and limitations of each respectively

7.3 Comparison of our Method against Current Issues

In this section, the comparison of our proposed methods in ea h step will be compared with the current issues. First, the comparison of our feature extraction and classification method against the method proposed by G. Roth and M. D. Levine for extracting geometric primitives [Ref. 7.1] will be made. Second, our method of feature matching against the geometric primitive matching will be made. Third, our method of grouping edge points into feature numbers will be compared against the method proposed by James N. H. and Jezekiel B. A. [Ref. 7.2]. Finally, the comparison of the method of object matching will be presented.

7.3.1 The Comparison of Feature Extraction Methods

In our proposed approach, edge detection is performed and the resulting edge map is used for feature extraction. The features are localized and being encoded into an integer ranging from 0 to 255. No statistical method is applied, no ambiguity is present, no floating point calculation is needed and all the processing elements can be processed at the same time. If the maximum parallel hardware is present, all the classification processes can be finished in one time interval. Thus real time processing for real world images should have no problem. The disadvantages of our proposed method for feature classification include : (1) there is no global feature consideration, (2) there is no object clustering ability, and (3) it contains no information for higher level objects.

In the primitive extraction proposed in [Ref. 7.1], the advantages of the method include the capability of extracting a wide variety of different geometric primitives and can use many different types of cost functions. It does not require any ordering of the geometric data . It can also be applied to both sparse and dense, single and multi-view geometric data. The geometric primitives are statistically optimal, and it can be formulated by using a parallel machine for faster processing. Its results are represented by a set of mathematical equations and can be thought of containing some high level information of an object.

The main limitations of the methods proposed in [Ref. 7.1] are that each time, only a single primitive can be extracted. It needs comparably heavy computer time when compared with our approach. The features are in higher level which may make the comparison of two similar feature more complicated. And vitally, there is no guaranty for the computation time which may cause real time processing impossible and hence is not selected for our proposed model.

7.3.2 The Comparison of Feature Matching Methods

The method for matching two features in our proposed model is by using an EXCLUSIVE OR operator which is very common and efficient. It also provides inexact matching by counting the number of bits set in the resulting integer. For example, ***c*** = ***a*** EXCLUSIVE OR ***b***, where ***a*** and ***b*** are two features being compared, if all the bits in ***c*** are 0, then ***a*** and ***b*** are completely the same. The number of bits set in ***c*** indicates the number of patterns which are different. Thus the result ***c*** can act as an indicator for the degree of matching of ***a*** and ***b***.

The method for matching two geometric primitives, such as the one described in [Ref. 7.1], involves complicated operations like statistical calculation, expansion of series and etc.(Comparison of two high level descriptors which are not the same may not be easy.). Although it can still provide inexact matching indication, it is difficult to give a fair description for it (it is difficult to generate a fair quantitative description.).

7.3.3 The Comparison of Feature Grouping Strategies

The feature grouping strategy in our proposed method is performed in pixel level and involves no high level knowledge. This classification method is simple and easy to perform (which is completed by the convolution of a classification mask [Ref.

7.4] with the edge map obtained). The features of the whole image frame can be completed in a more or less constant time limit.

The feature grouping strategy proposed in [Ref. 7.2] uses circular arcs, straight line segments combined with statistical methods to give a quantitative description of features. The calculation is complicated and there is no guaranty for time limit for the completion of the grouping process. And this approach has the same problems as described in section 7.3.3 when carry out feature comparison in two image frames (especially the comparison of two similar features but their descriptors are different).

7.3.4 The Comparison of Object Matching Methods

The object matching method proposed in this thesis uses the segmented image to account for the low level feature matching results for the calculation of matching strength. The matching decision is made according to the matching strength obtained. The appearance of an object can be determined from the homogeneous region of the segmented image (any image segmentation algorithm such as the approaches described in [Ref. 7.3] can be used if it can give a cut of homogeneous region which occurs in the boundaries of the related object.). This method, although may have errors, it will not cause serious consequences as the errors are limited to 1 or 2 units. The general trend of the motion of an object will not be misinterpreted because its decision is supported by both of the high level object matching strength and low level feature matching results.

The object matching methods such as area correlation, least square, best fit and etc., have their own advantages and disadvantages when compared with our method. The advantages include that they are statistically optimal, the processing is carried out in high level directly which does not need low level information, etc.. The disadvantages include that statistically optimal does not mean it is the real situation, great and serious errors may occur in some situations while the condition(s) of statistically optimum can still be maintained, it is in lack of real situation data support.

7.4 Discussion and Conclusion

From the experiments presented in last section, we can see that the results obtained are impressive. We believe that there is no comparable result which are published so far. However, careful observations may lead to raise questions. First of all, what is the problem with the results obtained in Figure 8c and 8d ?

According to Figure 7.3a, the IC is of one object, why the results show two motion quantities [2,1] and [2,2]? There are two answers to this question. This first one is that there are too many features for matching, the value $SM_i[dx,dy]$ for mismatch is stronger than the correct match. Therefore, when we carry out object resolution algorithm, the object is misresolved. The second answer may due to the equipment error when taking the pictures. Since the pictures are digitized at different time intervals, the truncation error from the equipment may result in the shift of the object either to the left or to the right, either to the top or to the bottom for 1 or 2 pixels. Thus the object is *detached*. It appears in the image map as different parts.

Another question is that: "does the success of detection depends on the quality of the image segmentation algorithm?". The answer is *YES*. The success of detection not merely depends on the image segmentation algorithm, but also on edge detection and feature classification. The main reason for the dependency on image segmentation is the calculation of the *matching strength* of the region. If the result of segmentation is poor, then the matching strength can be miscalculated and hence the object resolution algorithm can be misused, and finally leads to a wrong result.

For the above mentioned problems, the errors due to the inaccuracy of the hardware equipment seem not avoidable and have no way for improvement. However, the other two problems can be improved. First, the stronger mismatch $SM_i[dx,dy]$ can be improved by controlling the mismatch features in the initial stage of action in layer 4. Second, the error due to the calculation of *matching strength* of the region because of the poor performance of the image segmentation algorithm can be improved by improving the image segmentation algorithm. However, this is not easy, there are still many problems in image segmentation need to be solved. So far, no good segmentation algorithm has been developed.

Finally we can conclude that the translation detection of rigid body can be solved with acceptable results without much difficulty by the implementation of the theory and algorithms described in this thesis, perhaps with some improvement. However, the detection of all types of motion of a rigid body may still have a long way to go and may not be easy. By all types of motion, we mean that an object has performed one or more of the following types of motion or any of the combination of them, the motion types are :

(1) Translation ;
(2) Rotation ;
(3) Revolution .

The solution for these types of motions may involve the need for a large quantity of core memory, the need of great computation power and the development of new algorithms, etc.. The implementation of the existing algorithms in new and complex parallel computers with a large number of processing elements may be a trend to solve the motion problem.

Summer, 1993

Appendix A Connection Between Layer1 and Layer2

The cell processing action in the connection between Layer 1 and Layer 2 using Laplacian Operator. This program takes the input of 5 parameters. They are

[1] Image bit map filename ;
[2] Output edge filename ;
[3] Threshold value ;
[4] The number of rows of the image, and
[5] The number of columns of the image.

The Usage format is :

```
ProgramName [1] [2] [3] [4] [5]
```

/*** Source code for an example connection using Laplacian Operator ***/

```
# include "GVAR.H"
FILE *f1;
FILE *f2;
BYTE edge[cmax];
BYTE huge img[cmax][rmax];
int cl,rw;


/************** Program Subroutine Definition *************/
/****************** Laplacian Edge Detector
****************/
void compute_edge1(register int fsh)
{
```

```
register int m,n;
register int sum;

/** Initialization of edge **/
printf("Computing Edge  Please wait!");
edge[0] = edge[cl-1] = 255 ;
for (m=1;m<=rw-2;m++)
{
 for (n=1;n<=cl-2;n++)
 {
  /******************** Laplacian Operator **************/
  /*

            -1 -1 -1
            -1  8 -1
            -1 -1 -1
  */
  sum = 8*img[m][n]-img[m-1][n-1]-img[m-1][n]-img[m-1][n+1]
;
  sum=sum-img[m][n-1]-img[m][n+1]-img[m+1][n-1]-
img[m+1][n];
  sum = sum-img[m+1][n+1] ;
  /* printf("[%d] ",sum); */
  if (sum >= fsh)
  {
   edge[n] = 255 ;
  }
  else
  {
   edge[n] = 0 ;
  }
  }
  fwrite(&edge,1,rw,f2);
  /** End for n **/
 }
/** End for m **/
 return NULL;
}
```

```
/************* End of Compute Edge program *************/

main(int argc, char *argv[])
{
 register int i,j,fsh;

 if (argc < 6)
 {
  printf("Invalid Parameter,
Usage{%s}{image}{edge}{threshold}
          {row} {col}",argv[0]);
 }
 else
 {
  fsh = atoi(argv[3]);
  rw = atoi(argv[4]);
  cl = atoi(argv[5]);
  f1 = fopen(argv[1],"rb");
  for (i=0;i<(rw-1);i++)
  {
   fread(&edge,1,cl,f1);
   for (j=0;j<(cl-1);j++)
   {
    img[i][j] = edge[j];
    /* printf("[%d] ",img[i][j]); */
   }
  }
  fclose(f1);
  f2 = fopen(argv[2],"wb");
  printf("[%d] ",sizeof(edge));
  for (j=0;j<(cl-1);j++)
    edge[j] = 255;
  fwrite(&edge,1,rw,f2);
  compute_edge1(fsh);
  for (j=0;j<(cl-1);j++)
    edge[j] = 255;
  fwrite(&edge,1,rw,f2);
  fclose(f2) ;
```

```
        }
      }  /* End of main program */
```

Appendix B Connection Between Layer 2 and Layer 3

The cell processing action in the connection between Layer 1 and Layer 2 using a 3 x 3 classifier. This program takes the input of 4 parameters. They are

[1] Edge bit map filename;
[2] Ouput edge attribute filename ;
[3] The number of rows of the image, and
[4] The number of columns of the image.

The Usage format is :

```
ProgramName [1] [2] [3] [4]
```

/*** Source code for an example connection using a 3 x 3 classifier ***/

```
# include "GVAR.H"
FILE *f1;
FILE *f2;
BYTE att[cmax];
BYTE huge edge[cmax][rmax];
int cl,rw;


/************ Program Subroutine Definition *************/
void compute_att()
{
 register int m,n,i,j;
 register int sum;
```

```
/**************** Initialization of edge ****************/
printf("Computing attribute!  Please wait!");
att[1] = att[cl-2] = 0 ;
att[0] = att[cl-1] = 255 ;
for (m=2;m<=rw-3;m++)
{
 for (n=2;n<=cl-3;n++)
 {
  sum = 0 ;

  /************** Attribute evaluation ****************/
  if (edge[m][n] == 0)
  {

   /**************** Non-edge Point *****************/
   att[n] = 0 ;
  }
  else
  {

   /************* Evaluating Upper Row **************/
   if (edge[m-1][n-1] != 0) {
    sum = sum + 1 ;
   }
   if (edge[m-1][n] != 0) {
    sum = sum + 2 ;
   }
   if (edge[m-1][n+1] != 0) {
    sum = sum + 4 ;
   }

   /************* Evaluating Middle Row **************/
   if (edge[m][n-1] != 0) {
    sum = sum + 8 ;
   }
   if (edge[m][n+1] != 0) {
    sum = sum + 16 ;
```

```
      }

      /*************** Evaluating bottom row **************/
      if (edge[m+1][n-1] != 0) {
        sum = sum + 32 ;
      }
      if (edge[m+1][n] != 0) {
        sum = sum + 64 ;
      }
      if (edge[m+1][n+1] != 0) {
        sum = sum + 128 ;
      }

      switch(sum) {
        case 129 : sum = 0 ;
                   break ;
        case 66  : sum = 0 ;
                   break ;
        case 36  : sum = 0 ;
                   break ;
        case 24  : sum = 0 ;
                   break ;
        case 255 : sum = 0 ;
                   break ;
      } /*** switch ***/
      att[n]  = sum ;
     } /*** else ***/

     /************* End of attribute evaluation **************/
    } /*** for n ***/
    fwrite(&att,1,rw,f2);
  } /** End for m **/
  return NULL;
}

/************ End of Compute Edge program *************/

main(int argc, char *argv[])
```

```
{
 register int i,j,fsh;

 if (argc < 5)
 {
  printf("Invalid Parameter, Usage {%s} {edge} {attribute}
{row}
                                        {col}",argv[0]);
 }
 else
 {
  rw = atoi(argv[3]);
  cl = atoi(argv[4]);
  f1 = fopen(argv[1],"rb");
  for (i=0;i<(rw-1);i++)
  {
   fread(&att,1,cl,f1);
   for (j=0;j<(cl-1);j++)
   {
    edge[i][j] = att[j];
    /* printf("[%d] ",edge[i][j]); */
   }
  }
  fclose(f1);
  f2 = fopen(argv[2],"wb");
  for (j=0;j<(cl-1);j++)
    att[j] = 255;
  fwrite(&att,1,rw,f2);
  for (j=1;j<(cl-2);j++)
    att[j] = 0;
    att[0]=att[cl-1] = 255 ;
  fwrite(&att,1,rw,f2);
  compute_att();
  for (j=1;j<(cl-2);j++)
    att[j] = 0;
    att[0]=att[cl-1] = 255 ;
  fwrite(&att,1,rw,f2);
  for (j=0;j<(cl-1);j++)
```

```
      att[j] = 255;
    fwrite(&att,1,rw,f2);
    fclose(f2) ;
   }
 }  /* End of main program */
```

Appendix C Conection Between Layer3 and Layer4 and self iteration of Layer 4.

The cell processing action in the connection between Layer 3 and Layer 4 and then the cells in Layer 4 performs self iteration. This program takes the input of 2 parameters. They are

[1] Attribute bit map filename at time = t_n;

[2] Attribute bit map filename at time = t_{n+1};

The Usage format is :

```
ProgramName [1] [2]
```

/*** Source code for an example connection using Laplacian Operator ***/

```
/*********************************************************************/
/* Program Name :   MATCH                                             */
/* Function     :   To build translation correspondence from edge and */
/*                  edge attribute characteristic table               */
/* Author       :   Alex, Wong Hin Lau                                */
/*********************************************************************/

    /*************** Globle constant Definition ***************/
    # include "GVAR.H"
    # define maxcll 256
    # define x14    64
    # define times  4
    # define celnum 8192
    # define maxele 65536
    # define maxnum 255
# define maxlvl 25
# define maxx  6
# define maxy  6
# define maxatt 2000
# define HIGH  255
# define LOW   0
# define WIN   3
# define maxcode  62
# define EPSILON 10

/************** Globle Variable Definition ***************/

FILE *f1;
FILE *f2;
BYTE        list[celnum][2];
unsigned int  score[maxcode] ;
unsigned int  ptr,ptr1,ptr2;
BYTE huge tmplist[maxele][2] ;
BYTE tmpimg[x14][maxcll] ;
BYTE huge img1[maxcll][maxcll] ;
/****
   huge img2[maxcll][maxcll] ;
BYTE huge edg1[maxcll][maxcll] ,
   huge edg2[maxcll][maxcll] ;
****/
BYTE huge att1[maxcll][maxcll] ,
   huge att2[maxcll][maxcll] ;
BYTE huge cell[maxcll][maxcll] ;
BYTE huge map1[maxcll][maxcll] ,
   huge map2[maxcll][maxcll] ;
char mf1[12], mf2[12] ;

/*************** Output filenames definition **************/
char *fname[maxy][maxx] = {
          "lz00","lz01","lz02","lz03","lz04","lz05",
          "lz10","lz11","lz12","lz13","lz14","lz15",
          "lz20","lz21","lz22","lz23","lz24","lz25",
          "lz30","lz31","lz32","lz33","lz34","lz35",
                "lz40","lz41","lz42","lz43","lz44","lz45",
                "lz50","lz51","lz52","lz53","lz54","lz55",
                        };

/****************************/

char *fname[maxy][maxx] = {
                "lz00","lz01","lz02","lz03","lz04",
                    "lz05","lz06","lz07","lz08",
                    "lz09","lz0A",
                "lz10","lz11","lz12","lz13","lz14",
                    "lz15","lz16","lz17","lz18",
                    "lz19","lz1A",
                "lz20","lz21","lz22","lz23","lz24",
                    "lz25","lz26","lz27","lz28",
                    "lz29","lz2A",
                "lz30","lz31","lz32","lz33","lz34",
                    "lz35","lz36","lz37","lz38",
                    "lz39","lz3A",
                "lz40","lz41","lz42","lz43","lz44",
                    "lz45","lz46","lz47","lz48",
                    "lz49","lz4A",
                "lz50","lz51","lz52","lz53","lz54",
                    "lz55","lz56","lz57","lz58",
                    "lz59","lz5A",
                "lz60","lz61","lz62","lz63","lz64",
                    "lz65","lz66","lz67","lz68",
                    "lz69","lz6A",
                "lz70","lz71","lz72","lz73","lz74",
                    "lz75","lz76","lz77","lz78",
                    "lz79","lz7A",
                "lz80","lz81","lz82","lz83","lz84",
                    "lz85","lz86","lz87","lz88",
                    "lz89","lz8A",
                "lz90","lz91","lz92","lz93","lz94",
                    "lz95","lz96","lz97","lz98",
                    "lz99","lz9A",
                "lzA0","lzA1","lzA2","lzA3","lzA4",
                    "lzA5","lzA6","lzA7","lzA8",
                    "lzA9","lzAA",
                };

****************************/

/************* Program Subroutine Definition *************/

void init_map()
{
 register int i,j;

 for (i=0;i<=maxnum;i++)
 {
  for (j=0;j<=maxnum;j++)
  {
   map1[i][j] = HIGH ;
   map2[i][j] = LOW  ;
  }
 }
}

void init_map1()
{
 register int i,j;

 for (i=0;i<=maxnum;i++)
 {
  for (j=0;j<=maxnum;j++)
  {
   map1[i][j] = LOW  ;
  }
 }
}

void init_map2()
{
 register int i,j;
 for (i=0;i<=maxnum;i++)
 {
  for (j=0;j<=maxnum;j++)
  {
   map2[i][j] = LOW  ;
  }
 }
}


void init_cell()
{
 register int i,j;

 /********* Initializing attribute match variables ************/
 /****
 printf("Initializing cell ..... ");
 ****/
 for (i=0;i<=maxnum;i++)
 {
  for (j=0;j<=maxnum;j++)
  {
   if ((i==0)||(i==maxnum))
   {
    cell[i][j] = HIGH ;  /** Init. Boundary to HIGH **/
   }
   else
   {
    cell[i][j] = LOW ;
   }
  }
  cell[i][0] = cell[i][maxnum] = HIGH ; /** Init. Boundary to
HIGH **/
 }
 /****
 printf("Initializing cell completed ! \n");
 ****/
```

```
}
/*************** End of procedure init_cell ***************/

/******** This subroutine opens all related image files ********/

void read_images(char *fname1,char *fname2, BYTE ch)
{
 char *img = ".bmp";
 char *edg = ".l2";
 char *att = ".at";
 char fimg1[12], fimg2[12] ;
 char fedg1[12], fedg2[12] ;
 char fatt1[12], fatt2[12] ;
 char fn1[12], fn2[12] ;
 register int i,j,k ;

 /****
 printf("[%s] and [%s] \n", fname1, fname2) ;
 printf("Opening image files ......\n") ;
 ****/
 strcpy(fimg1,fname1) ;
 strcpy(fedg1,fname1) ;
 strcpy(fatt1,fname1) ;

 strcpy(fimg2,fname2) ;
 strcpy(fedg2,fname2) ;
 strcpy(fatt2,fname2) ;

 strncat(fimg1,img,4) ;
 strncat(fimg2,img,4) ;
 strncat(fedg1,edg,3) ;
 strncat(fedg2,edg,3) ;
 strncat(fatt1,att,3) ;
 strncat(fatt2,att,3) ;
 switch(ch)
 {
  case 0 : /**** printf("Reading Bip Map Images.\n"); ****/
        strcpy(fn1,fimg1) ;
```

```
            strcpy(fn2,fimg2) ;
            break ;
    case 1 : /***** printf("Reading Edge Images.\n"); ****/
            strcpy(fn1,fedg1) ;
            strcpy(fn2,fedg2) ;
            break ;
    case 2 : /***** printf("Reading Attribute Images.\n"); *****/
            strcpy(fn1,fatt1) ;
            strcpy(fn2,fatt2) ;
            break ;
    default: printf("Warning : Default Image Readimg Entered ! \n")
;
            strcpy(fn1,fatt1) ;
            strcpy(fn2,fatt2) ;
            break ;
  }

/*****
printf("[%s] [%s] [%s] [%s] [%s] [%s] \n",
     fimg1,fimg2,fedg1,fedg2,fatt1,fatt2) ;
*****/

 /********* Open Argument1 Attribute Map Image **********/
  if ((f2 = fopen(fn1,"rb")) == NULL)
  {
   printf("Open [%s] for Read not successful ! \n",fn1);
   exit(0) ;
  }
  for (i=0; i<= times-1; i++)
  {
   fread(&tmpimg,1,sizeof(tmpimg),f2);
   for (j=0; j<=x14-1; j++)
   {
    for (k=0; k<=maxcll-1; k++)
    {
     att1[i*64+j][k] = tmpimg[j][k] ;
    }
   }
 }
 fclose(f2);

/********** End of Opening Argument 1 Files *************/

/********* Open Argument2 Attribute Map Image **********/

 if ((f2 = fopen(fn2,"rb")) == NULL)
 {
  printf("Open [%s] for Read not successful ! \n",fn2);
  exit(0) ;
 }
 for (i=0; i<= times-1; i++)
 {
  fread(&tmpimg,1,sizeof(tmpimg),f2);
  for (j=0; j<=x14-1; j++)
  {
   for (k=0; k<=maxcll-1; k++)
   {
    att2[i*64+j][k] = tmpimg[j][k] ;
   }
  }
 }
 fclose(f2);

/********** End of Opening Argument 1 Files *************/
}


/*********** End of Open Images Procedures **************/

void read_segment_image()
{
 register int i,j,k ;

 if ((f2 = fopen("test.seg","rb")) == NULL)
 {
  printf("Open [test.seg] for Read not successful ! \n");
```

```
   exit(0) ;
  }
  for (i=0; i<= times-1; i++)
  {
   fread(&tmpimg,1,sizeof(tmpimg),f2);
   for (j=0; j<=x14-1; j++)
   {
    for (k=0; k<=maxcll-1; k++)
    {
     img1[i*64+j][k] = tmpimg[j][k] ;
    }
   }
  }
  fclose(f2);
}

void mark_region()
{
  register BYTE x,y;
  register BYTE gl;

  ptr1 = ptr1 - 1 ;
  x = tmplist[ptr1][0] ;
  y = tmplist[ptr1][1] ;
  gl = img1[x][y] ;

  if (map1[x][y] == LOW)
  {
   map1[x][y] = HIGH;
  }

  if ((img1[x-1][y-1] == gl) && (map1[x-1][y-1] == LOW))
  {
   map1[x-1][y-1] = HIGH;
   tmplist[ptr1][0] = (x-1) ;
   tmplist[ptr1][1] = (y-1) ;
   ptr1 = ptr1 + 1 ;
  }
```

```
if ((img1[x-1][y] == gl) && (map1[x-1][y] == LOW))
{
 map1[x-1][y] = HIGH;
 tmplist[ptr1][0] = (x-1) ;
 tmplist[ptr1][1] = y ;
}

if ((img1[x-1][y+1] == gl) && (map1[x-1][y+1] == LOW))
{
 map1[x-1][y+1] = HIGH;
 tmplist[ptr1][0] = (x-1) ;
 tmplist[ptr1][1] = (y+1) ;
 ptr1 = ptr1 + 1 ;
}

if ((img1[x][y-1] == gl) && (map1[x][y-1] == LOW))
{
 map1[x][y-1] = HIGH;
 tmplist[ptr1][0] = x ;
 tmplist[ptr1][1] = (y-1) ;
 ptr1 = ptr1 + 1 ;
}

if ((img1[x][y+1] == gl) && (map1[x][y+1] == LOW))
{
 map1[x][y+1] = HIGH;
 tmplist[ptr1][0] = x ;
 tmplist[ptr1][1] = (y+1) ;
 ptr1 = ptr1 + 1 ;
}

if ((img1[x+1][y-1] == gl) && (map1[x+1][y-1] == LOW))
{
 map1[x+1][y-1] = HIGH;
 tmplist[ptr1][0] = (x+1) ;
 tmplist[ptr1][1] = (y-1) ;
 ptr1 = ptr1 + 1 ;
```

```
  }

  if ((img1[x+1][y] == gl) && (map1[x+1][y] == LOW))
  {
   map1[x+1][y] = HIGH;
   tmplist[ptr1][0] = (x+1) ;
   tmplist[ptr1][1] = y ;
   ptr1 = ptr1 + 1 ;
  }

  if ((img1[x+1][y+1] == gl) && (map1[x+1][y+1] == LOW))
  {
   map1[x+1][y+1] = HIGH;
   tmplist[ptr1][0] = (x+1) ;
   tmplist[ptr1][1] = (y+1) ;
   ptr1 = ptr1 + 1 ;
  }
}

/************* BUild Segment Image List *****************/

void build_segment_list()
{
  register int i,j;

  printf("Building Object List ......\n");
  ptr = 0 ;
  for (i=0;i<=maxnum;i++)
  {
   for (j=0;j<=maxnum;j++)
   {
    if (map1[i][j] == LOW)
    {
     list[ptr][0] = i ;
     list[ptr][1] = j ;
     ptr = ptr + 1 ;
     ptr1 = 0 ;
     tmplist[ptr1][0] = i ;
```

```
        tmplist[ptr1][1] = j ;
        printf("[%d][%d] ",i,j);
        ptr1 = ptr1 + 1 ;
        do
        {
         mark_region() ;
        } while (ptr1 > 0) ;
       }
      }
     }
     ptr = ptr - 1 ;
     printf("Building Object List Completed!\n");
     printf("There are [%d] objects.\n",ptr+1) ;
    }

    /************** Subroutine Write One Image **************/

    void write_image(char *fn)
    {
     register int i,j,k ;

     /****
     printf("Writing Image file [%s] \n",fn) ;
     ****/
     if ((f2 = fopen(fn,"wb")) == NULL)
     {
      printf("Open [%s] for Write not successful ! \n",fn);
      exit(0) ;
     }
     for (i=0; i<= times-1; i++)
     {
      for (j=0; j<=x14-1; j++)
      {
       for (k=0; k<=maxcll-1; k++)
       {
       tmpimg[j][k] = cell[i*x14+j][k] ;
       }
      }
    fwrite(&tmpimg,1,sizeof(tmpimg),f2);
  }
  fclose(f2);
}

/********** Procedure to calculate matching results **********/

BYTE match_results(a, b)
register BYTE a,b;
{
  register BYTE d ;

  if (a==b)
  {
    return 255 ;
  }
  switch(a)
  {
    case 1   : if ((b==2)||(b==8))
                { return 128; }
               else
                { return 0; }
    case 2   : if ((b==1)||(b==4))
                { return 128; }
               else
                { return 0; }
    case 4   : if ((b==2)||(b==16))
                { return 128; }
               else
                { return 0; }
    case 8   : if ((b==1)||(b==32))
                { return 128; }
               else
                { return 0; }
    case 16  : if ((b==4)||(b==128))
                { return 128; }
               else
                { return 0; }
  case 32  : if ((b==8)||(b==64))
             { return 128; }
            else
             { return 0; }
  case 64  : if ((b==32)||(b==128))
             { return 128; }
            else
             { return 0; }
  case 128 : if ((b==16)||(b==64))
             { return 128; }
            else
             { return 0; }
  default  : d = a^b ;
            switch(d)
            {
             case 1   : return 128;
             case 2   : return 128;
             case 4   : return 128;
             case 8   : return 128;
             case 16  : return 128;
             case 32  : return 128;
             case 64  : return 128;
             case 128 : return 128;
             default  : return 0  ;
            }
 }
}

/************ end of procedure match_results **************/

/******* This Procedure Performs Subset illumination ********/

void subset_illuminate(BYTE a1, BYTE a2)
{
 register BYTE i,j,x,y,sum,t1;

 /********* Subset Illimination By a 5x5 matrix *************/
 t1 = WIN + 1 ;
 for (i=WIN;i<=(maxnum-t1);i++)
 {
  for (j=WIN;j<=(maxnum-t1);j++)
  {
   if (cell[i][j] > 0)
   {
    sum = 0 ;
    for (x=i-WIN;x<=i+WIN;x++)
    {
     for (y=j-WIN;y<=j+WIN;y++)
     {
      if (cell[x][y] == 255)
      {
       sum = sum + 3 ;
      }
      else if (cell[x][y] == 128)
      {
       sum = sum + 3 ;
      }
     }
    }
    if (sum > map2[i][j]) /**** Update Map2 and Map1****/
    {
     map2[i][j] = sum ;
     map1[i][j] = (a1*11)+a2+1 ;
    }
   }
  }
 }
}

/********* End of Procedure Subset Illumination ************/

/*********** Building edge from attribute table *************/

void build_correspondence()
{
 register BYTE i,j,x,y;
 register BYTE tmp1,tmp2 ;

 printf("Building Edge from attribute characteristic ......\n");

 /******* if you want to read this part, read it carefully *******/

 read_images(mf1,mf2,2);
 for (y=0;y<=maxy-1;y++)
 {
  for (x=0;x<=maxx-1;x++)
  {
   init_cell();
   for (i=1;i<=(maxnum-1-y);i++)
   {
    for (j=1;j<=(maxnum-1-x);j++)
    {
     tmp1 = att1[i][j] ;
     tmp2 = att2[i-y][j+x] ;
     if ((tmp1 != 0)&&(tmp2 != 0))
     {
      cell[i][j] = match_results(tmp1,tmp2) ;
     }
    }
   }
   subset_illuminate(y,x) ;
   /****
   write_image(fname[y][x]);
   ****/
  }
 }
}

/*********** End of procedure building_edge **************/

void init_score()
{
 register int i;

 for(i=0;i<=maxcode;i++)
 {
  score[i] = 0 ;
 }
}

void trace_region()
{
 register BYTE x,y;
 register BYTE gl;

 ptr1 = ptr1 - 1 ;
 x = tmplist[ptr1][0] ;
 y = tmplist[ptr1][1] ;
 gl = img1[x][y] ;
 map2[x][y] = 120 ;
 score[map1[x][y]] = score[map1[x][y]]+1 ;

 if ((img1[x-1][y-1] == gl) && (map2[x-1][y-1] == LOW))
 {
  map2[x-1][y-1] = 120;
  tmplist[ptr1][0] = (x-1) ;
  tmplist[ptr1][1] = (y-1) ;
  score[map1[x-1][y-1]] = score[map1[x-1][y-1]]+1 ;
  ptr1 = ptr1 + 1 ;
 }

 if ((img1[x-1][y] == gl) && (map2[x-1][y] == LOW))
 {
  map2[x-1][y] = 120;
  tmplist[ptr1][0] = (x-1) ;
  tmplist[ptr1][1] = y ;
  score[map1[x-1][y]] = score[map1[x-1][y]]+1 ;
  ptr1 = ptr1 + 1;
 }
```

```
if ((img1[x-1][y+1] == gl) && (map2[x-1][y+1] == LOW))
{
 map2[x-1][y+1] = 120;
 tmplist[ptr1][0] = (x-1) ;
 tmplist[ptr1][1] = (y+1) ;
 score[map1[x-1][y+1]] = score[map1[x-1][y+1]]+1 ;
 ptr1 = ptr1 + 1 ;
}

if ((img1[x][y-1] == gl) && (map2[x][y-1] == LOW))
{
 map2[x][y-1] = 120;
 tmplist[ptr1][0] = x ;
 tmplist[ptr1][1] = (y-1) ;
 score[map1[x][y-1]] = score[map1[x][y-1]]+1 ;
 ptr1 = ptr1 + 1 ;
}

if ((img1[x][y+1] == gl) && (map2[x][y+1] == LOW))
{
 map2[x][y+1] = 120;
 tmplist[ptr1][0] = x ;
 tmplist[ptr1][1] = (y+1) ;
 score[map1[x][y+1]] = score[map1[x][y+1]]+1 ;
 ptr1 = ptr1 + 1 ;
}

if ((img1[x+1][y-1] == gl) && (map2[x+1][y-1] == LOW))
{
 map2[x+1][y-1] = 120;
 tmplist[ptr1][0] = (x+1) ;
 tmplist[ptr1][1] = (y-1) ;
 score[map1[x+1][y-1]] = score[map1[x+1][y-1]]+1 ;
 ptr1 = ptr1 + 1 ;
}

if ((img1[x+1][y] == gl) && (map2[x+1][y] == LOW))
{
   map2[x+1][y] = 120;
   tmplist[ptr1][0] = (x+1) ;
   tmplist[ptr1][1] = y ;
   score[map1[x+1][y]] = score[map1[x+1][y]]+1 ;
   ptr1 = ptr1 + 1 ;
  }

  if ((img1[x+1][y+1] == gl) && (map2[x+1][y+1] == LOW))
  {
   map2[x+1][y+1] = 120;
   tmplist[ptr1][0] = (x+1) ;
   tmplist[ptr1][1] = (y+1) ;
   score[map1[x+1][y+1]] = score[map1[x+1][y+1]]+1 ;
   ptr1 = ptr1 + 1 ;
  }
 }

BYTE check_maximum()
{
 register BYTE i,j,k;

 j = 0 ;
 k = 1 ;
 for (i=0;i<= maxcode-1;i++)
 {
  if (score[i] > j)
  {
   j = score[i] ;
   k = i ;
  }
 }
 return k ;
}

void identify_object(BYTE maxscore)
{
 register BYTE x,y;
 register BYTE gl;
```

```
ptr1 = ptr1 - 1 ;
x = tmplist[ptr1][0] ;
y = tmplist[ptr1][1] ;
gl = img1[x][y] ;
map2[x][y] = HIGH ;
map1[x][y] = maxscore ;

if ((img1[x-1][y-1] == gl) && (map2[x-1][y-1] == 120))
{
 map2[x-1][y-1] = HIGH;
 map1[x-1][y-1] = maxscore;
 tmplist[ptr1][0] = (x-1) ;
 tmplist[ptr1][1] = (y-1) ;
 ptr1 = ptr1 + 1 ;
}

if ((img1[x-1][y] == gl) && (map2[x-1][y] == 120))
{
 map2[x-1][y] = HIGH;
 map1[x-1][y] = maxscore;
 tmplist[ptr1][0] = (x-1) ;
 tmplist[ptr1][1] = y ;
 ptr1 = ptr1 + 1;
}

if ((img1[x-1][y+1] == gl) && (map2[x-1][y+1] == 120))
{
 map2[x-1][y+1] = HIGH;
 map1[x-1][y+1] = maxscore;
 tmplist[ptr1][0] = (x-1) ;
 tmplist[ptr1][1] = (y+1) ;
 ptr1 = ptr1 + 1 ;
}

if ((img1[x][y-1] == gl) && (map2[x][y-1] == 120))
{
 map2[x][y-1] = HIGH;
 map1[x][y-1] = maxscore;
 tmplist[ptr1][0] = x ;
 tmplist[ptr1][1] = (y-1) ;
 score[map1[x][y-1]] = score[map1[x][y-1]]+1 ;
 ptr1 = ptr1 + 1 ;
}

if ((img1[x][y+1] == gl) && (map2[x][y+1] == 120))
{
 map2[x][y+1] = HIGH;
 map1[x][y+1] = maxscore;
 tmplist[ptr1][0] = x ;
 tmplist[ptr1][1] = (y+1) ;
 ptr1 = ptr1 + 1 ;
}

if ((img1[x+1][y-1] == gl) && (map2[x+1][y-1] == 120))
{
 map2[x+1][y-1] = HIGH;
 map1[x+1][y-1] = maxscore;
 tmplist[ptr1][0] = (x+1) ;
 tmplist[ptr1][1] = (y-1) ;
 ptr1 = ptr1 + 1 ;
}

if ((img1[x+1][y] == gl) && (map2[x+1][y] == 120))
{
 map2[x+1][y] = HIGH;
 map1[x+1][y] = maxscore;
 tmplist[ptr1][0] = (x+1) ;
 tmplist[ptr1][1] = y ;
 ptr1 = ptr1 + 1 ;
}

if ((img1[x+1][y+1] == gl) && (map2[x+1][y+1] == 120))
{
 map2[x+1][y+1] = HIGH;
 map2[x+1][y+1] = maxscore;
```

```
   tmplist[ptr1][0] = (x+1) ;
   tmplist[ptr1][1] = (y+1) ;
   ptr1 = ptr1 + 1 ;
  }
}

void object_identification()
{
 register int i,j;
 register BYTE maxscore ;

 printf("Identifying Objects....") ;
 init_map2();
 for(i=0;i<=ptr;i++)
 {
  init_score();
  ptr1 = 0 ;
  tmplist[ptr1][0] = list[i][0];
  tmplist[ptr1][1] = list[i][1];
  printf("Object[%d] [%d][%d] ",i,list[i][0],list[i][1]);
  ptr1 = ptr1 + 1 ;
  do
  {
   trace_region() ;
  } while (ptr1 > 0) ;
  maxscore = check_maximum() ;
  printf("Maxscore=[%d]..",maxscore);
  ptr1 = 0 ;
  tmplist[ptr1][0] = list[i][0];
  tmplist[ptr1][1] = list[i][1];
  ptr1 = ptr1 + 1 ;
  do
  {
   identify_object(maxscore) ;
  } while (ptr1 > 0) ;
 }
 printf("Identification Completed!\n");
}
```

```
/***** This Procedure decodes the motion data in MAP1 ******/
void decode_map1()
{
 register BYTE i,j,x,y;
 register BYTE tmp1,tmp2 ;

 printf("Decoding Map1 ......");
 for (y=0;y<=maxy-1;y++)
 {
  for (x=0;x<=maxx-1;x++)
  {
   for (i=1;i<=maxnum-1;i++)
   {
    for (j=1;j<=maxnum-1;j++)
    {
     if (map1[i][j] == (y*11+x+1))
     {
      cell[i][j] = HIGH ;
     }
     else
     {
      cell[i][j] = LOW ;
     }
    }
   }
   write_image(fname[y][x]);
  }
 }
 printf("Completed!\n");
}

/*********** End of Procedure decode Map 1 **************/
/************ The main program for Match ****************/

void main(int argc, char *argv[])
{
 register int i,j,k;
```

```
  if (argc < 3)
  {
   printf("Usage : {%s} {Image 1} {image 2}",argv[0]);
  }
  else
  {
   strcpy(mf1,argv[1]) ;
   strcpy(mf2,argv[2]) ;
   read_segment_image();
   init_map1();
   build_segment_list();
   init_map();
   build_correspondence();
   printf("The value of PTR is [%d]",ptr+1);
   object_identification();
   decode_map1();
   /************* To test filename integraty ***************/
   for (i=0;i<=maxy-1;i++)
   {
    for (j=0;j<=maxx-1;j++)
    {
     printf("[%s]",fname[i][j]);
    }
    printf("\n");
   }

   /************** End of filename testing ****************/

   /******
   printf("Read Image Finisfed ! \n");
   write_image(fname[20]);
   printf("Write Image Finisfed ! \n");
   ******/
  }
}
/**************** End of main program *****************/
```

Appendix D Algorithm by Rangarajan and Shah for building motion correspodences

```
For k = 2 to n-1 do
(a) Set up M, a ($m_k * m_{k+1}$) matrix, with the $m_k$ points from the kth
    frame along the row and $m_{k+1}$ points from the (k+1)th frame along
    the column.

(b) Let $M[i,j] = \delta(m_k X_p^{k-1} X_i^{k} X_j^{k+1})$, when $\Phi^{k-1}(p) = i$ ;

(c) if ($m_{k+1} < m_k$) then
      i. For a = 1 to $m_{k+1}$ do
           A. Identify the minimum $[l_j, j]$ in each column j of M
           B. Compute priority matrix B, such that $B[l_j,j] = \sum_{i=1\ i<>l_j}^{m} M_{(i\ j)}$
           C. Select $[l_j,j]$ with highest priority value $B[l_j,j]$, and make
              $\Phi^k(l_j) = j$
           D. Mask row $l_j$ and column j from M
      ii. identify $m_k - m_{k+1}$ points which correspondence has not been
          found. For those points create new points in frame k+1
          by extrapolating the correspondence from frame k-1 to k.
      iii. Set $m_{k+1} = n$
    else
      i. for a = 1 to $m_{k+1}$ do
           A. Identify the minimum element $[l_j,j]$ in each row of M
           B. Compute priority matrix B, such that
              $B[l_j,j] = \sum_{j=1\ j<>l_i}^{m} M_{(i\ j)} + \sum_{k=1\ k<>i}^{m} M_{(k\ li)}$ for each i
           C. Select $[l_j,j]$ pair with highest priority value
              $B[i,l_i]$, and make $\Phi^k(i) = l_i$
           D. Mask row i and column $l_i$ from M.
```

The points present in the frame $\mathbf{I}_k$ but are missing in $\mathbf{I}_{k+1}$ are redefined by extra-polation using the corresponding points in $\mathbf{I}_{k-1}$ and $\mathbf{I}_k$.

References

Ref. 1.1 *Anil K. Jain "Fundamentals of digital image processings", Chapter 1, Prentice-Hall Imternational, 1989.*

Ref. 1.2 *R. M. Haralick. "Statistical and Structural Approaches to Texture." Proc. IEEE 67 (May 1979) : 786-809.*

Ref. 1.3 *Anil K. Jain "Fundamentals of digital image processings", Chapter 9, Section 11, P395-P397, Prentice-Hall Imternational, 1989.*

Ref. 1.4 *B. S. Lipkin and A. Rosenfeld (eds). Picture Processing and Psychopictorics. New York: Academic Press, 1970.*

Ref. 1.5 *T. Pavlidis. Structural Patern Recognition. New York: Springer-Verlag, 1977.*

Ref. 1.6 *Anil K. Jain "Fundamentals of digital image processings", Chapter 9 Prentice-Hall Imternational, 1989.*

Ref. 1.7 *KALIVAS AND ALEXANDER A. SAWCHUK "A region matching motion estimation algorithm" Image Understanding Vol 54, No. 2, September, p. 309-324, 1991*

Ref. 1.8 *James N. Huddleston and Jezekiel Ben-Arie "Grouping Edgels into Structural Entities Using Circular Symmetry, the Distributed Hough Transform, and Probabilistic Non-accidentalness" Image Understanding Vol 57, No. 2, March, pp. 227-242, 1993*

Ref. 1.9 *J. Ben-Arie, "The probabilistic peaking effect of viewed angles and distances with application to 3-D object recognition" IEEE Trans. Pattern Analysis, Mach. Intell. 12, 1990, 760-774.*

Ref. 1.10 *J. Ben-Arie, "Probabilistic models of observed features and aspects with appilcationto weighted aspect graphs", Pattern Recogn. Lett. 11, 1990, 421-427*

Ref. 1.11 *R. Mohan and R. Nevatia, "Segmentation and description based on percetual organization", in IEEE CVPR COnf.,1989, pp. 333-341.*

Ref. 1.12 *J. Ben-Arie and J. N. Huddleston, "Grouping and forming quantitative descriptions of image features by a novel parallel algorithm", in SPIE Visual Communications and Image Processing Conf., Boston, MA, Nov., 1991, pp. 2-20.*

Ref. 1.13 *Y. Xia, "Skeletonization via the realization of the fore front's propagation and extinction in digital binary sharps", IEEE Trans. Pattern Anal. Mach. Intell. 11, 1989, 1076-1089.*

Ref. 1.14 *H. Rom and G. Medioni, "Hierarchical decomposition and axial representation of sharp, in SPIE Conf. on Stochastic Methods in Signal Processing and Computer Vision", San Diego, CA, JUly, 1991.*

Ref. 1.15 *Gerhard Roth and Martin D. Levine, "Exctracting Geometric Primitives", Image Understanding Vol 58, No. 1, July, pp. 1-22, 1993*

Ref. 1.16 *M. Rioux, "Laser rangefinders based on synchronized scanning", Appl., Opt. 23, 1985, 3837-3844.*

Ref. 1.17 *Y. Ohta and T. Kanada, "Stereo by intra- and Inter-scanline search using dynamic programming", IEEE Trans, Pattern Anal. Mach. Intell. 7, 1985, 139-154.*

Ref. 1.18 *P. J. Besl abd R. C. Jain, "Segmentation through variable-order surface fitting", IEEE Trans. Pattern Anal. Mach. Intell. 10, 1988, 167-192.*

Ref. 1.19 *D. Kim, J. Kim, P. Meer, D. Mintz, and A. Rosenfeld, "Robust Computer Vision : A least median of squares approach", in DARPA Image Understanding Workshop, May 1989, pp. 1117-1134.*

Ref. 1.20 D. Chen, "A data driven intermediate level feature extraction algorithm", IEEE Trans. Pattern Anal. Mach. Intell. 11, 1989, 749-758.

Ref. 1.21 F. Hampel, E. Ronchetti, P. Rousseeuw, and W. Stahel, "Robust Statistics : The approach based on influence functions", Wiley, New York, 1986.

Ref. 1.22 P. Huber, "Robust Statistics", Wiley, New York, 1981.

Ref. 1.23 Todd R. Reed and J. M. Hans Du Buf, "A review of Recent Texture Segmentation and Feature Extraction Techniques", Image Understanding Vol 57, No. 3, May, pp. 395-372, 1993

Ref. 1.24 K. I. Laws, "Texture image segmentation", Technical report USCIPI Report 940, Dept. of Elec. Eng., Image Processing Inst., University of Southern California , Los Angeles, January 1980.

Ref. 1.25 R. W. Conner, M. M. Trivedi, and C. A. Harlow, "Segmentation of a high-resolution urban scene using texture operators", Computer Vision Graphics Image Process. 25, 1984, 273-310.

Ref. 1.26 I. Dinstein, A. C. Fong, L. M. Ni, and K. Y. Wong, "Fast discrimination between homogeneous and textured regions", in Proceedings, 7th International Conference on Pattern Recognition, Montreal, Canada, July 30-August 2, 1984, pp. 361-363.

Ref. 1.27 M. Unser, "Local linear transformation for texture measurements", Signal Process. 11, 1986, 61-79.

Ref. 1.28 R. Wang, A. R. Hanson, and E. M. Riseman, "Texture analysis based on local standard deviation of intensity", IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Miami Beach, Florida, June 22-26, 1986, pp. 482-488.

Ref. 1.29 R. M. Haralick, "Image Segmentation, K. Shanmugam, and I. Dinstein, "Texture features for image classification", IEEE Trans. Systems Man Cybernet. 3(1), 1973, 610-621.

Ref. 1.30 *L. W. Abele, "Feature Selection by space invariant comparison with applications to the segmentation of texture pictures", in Proceedings of the 5th International Conference on Pattern Recognition, Miami Beach, Florida, December 1-4, 1980, pp. 535-539.*

Ref. 1.31 *G. Y. Xu and K. S. Fu, " Natural scene segmentation based on multiple threshold and texture measurement", in Proceedings, 7th International Conference on Pattern Recognition, Montreal, Canada, July 30-August 2, 1984, pp. 1213-1215.*

Ref. 1.32 *M. E. Jernigan and F. D' Astous, "Entropy-based texture analysis in the spatial frequency domain", IEEE Trans. Pattern Anal. Mach. Intell. 6(2), 1984, 237-243.*

Ref. 1.33 *Farshid Arman and J. K. Aggarwal, "CAD-based Vision : Object recognition in cluttered range images using recognition strategies", Image Understanding Vol 58, No. 1, July, pp. 33-48, 1993*

Ref. 1.34 *P. J. Besl and R. C. Jain, "Three-dimensional object recognition" , ACM Computer Surveys 17(1), 1985, 75-145*

Ref. 1.35 *F. Armen and J. K. Aggarwal, "Model-based object recognition in dense range images-A review", in ACM Computer Surveys, March, 1993*

Ref. 1.36 *W. E. L.Grimson, "On the recognition of parameterized objects", in 4th International Symposium on Robotics Research, Santa Cruz, CA., Aug. 1987*

Ref. 1.37 *W. E. L.Grimson, "On the recognition of curved objects", IEEE Trans. Pattern Anal. Mach. Intell. 11(6), 1989, 632-642.*

Ref. 1.38 *W. E. L.Grimson and T. Lozeno-Perez, "Localzing overlapping overlapping parts by searching the interpretation tree", IEEE Trans. Pattern Anal. Mach. Intell. 9(4), 1987, 469-482.*

Ref. 1.39 *P. J. Flynn and A. K. Jain, "Bonzai: 3-D object recognition using constrained search", in Third International Conference on Computer Vision, Osaka, Janpan, Dec. 4-7, 1990, pp. 263-267.*

Ref. 1.40 *T. J. Fan, G. Medioni and R. Nevatia, "Segmented descriptions of 3-D surfaces", IEEE Int. J. Rob. Autom. 3(6), 1987, 527-538.*

Ref. 1.41 *T. J. Fan, G. Medioni and R. Nevatia, ""Recognizing 3-D objects using surface descriptions", IEEE Trans. Pattern Anal. Mach. Intell. 11(11), 1989, 1140-1157.*

Ref. 1.42 *T. J. Fan, "Describing and Recognizing 3-D Objects Using Surface Properties", Springer-Verlag, New York, 1990*

Ref. 1.43 *C. Hansen and T. C. Henderson, "Towards automatic generation of recognition strategies", in Proceeding, International Conference on Computer Vision, Tampa, FL, Dec. 5-8, 1988, pp. 275-279.*

Ref. 1.44 *C. Hansen and T. C. Henderson, "CAGD-based computer vision", IEEE Trans. Pattern Anal. Mach. Intell. 11(11), 1989, 1181-1193.*

Ref. 1.45 *Alessandro Verri and Tomaso Poggio, "Motion Field and optical flow", IEEE Trans. Pattern Anal. Mach. Intell. 11(5), May, 1989, 490-498.*

Ref. 1.46 *E. C. Hildreth, "The measurement of visual motion", Cambridge ; MA: Mit press, 1984.*

Ref. 1.47 *-- "The computation of the velocity field", Proc. Roy. Soc. London, vol. B221, pp. 189-220, 1984*

Ref. 1.48 *A. M. Waxman , "Image flow theory", in Advanced in computer vision, C. Brown, Ed. Norwood , NJ, 1991.*

Ref. 1.49 *B. K. P. Horn , "Robot Vision", Cambridge , MA, MIT press/ Mc-Graw-Hill, 1986*

Ref. 1.50	*K. Kanatani, "Structure from motion without correspondence: General principles", in Proc. Image Understanding Workshop, Miami, FL, 1985, pp. 107-116.*
Ref. 1.51	*H. H. Nagel, "Recent advances in image sequence analysis", in Proc. Premier Colloque Image - Traitment, Synthese, Technologie et Applications, Biarritz, France, 1984, pp. 544-558.*
Ref. 1.52	*H. H. Nagel and W. Enkelmann, "An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences", IEEE Trans. Pattern Anal. Mach. Intell. vol. PAMI-8, 1986, pp. 565-593.*
Ref. 1.53	*A. Verri and T. Poggio, "Against quantitative optical flow", in Proc. 1st ICCV, London, June 1987*
Ref. 1.54	*A. Verri, F. Girosi, and V. Torre, "Mathematical properties of the 2-D motion field: From singular points to motion parameters", J. Opt. Soc. Amer., 1989*
Ref. 1.55	*S. Uras, F. Girosi, A. Verri, and V. Torree, "Computational approach to motion perception", Biol. Cybern., vol. 60, pp. 79-87, 1988*
Ref. 1.56	*Gilad Adiv, "Determining three-dimensional motion structure from optical flow generated by several moving objects", IEEE Trans. Pattern Anal. Mach. Intell. vol. PAMI-7, Number 4, July, 1986, pp. 384-401.*
Ref. 3.1	*J. M. S. Prewitt "Object Enhancement and Extraction". Picture Processing and Psychopictorics. New York; Academic Press, 1970.*
Ref. 3.2	*L. S. Davis. "A Survey of Edge Detection Techniques". Computer Graphics and Image Processing, vol. 4, pp. 248-270, 1975.*

Ref. 3.3 *A. Rosenfeld and M. Thurston. "Edge and Curve Detection for Visual Scene Analysis". Computer Methods in image processing. Los Angeles; IEEE Computer Society, 1977.*

Ref. 3.4 *L. G. Roberts. "Machine Perception of Three Dimensional Solids". Computer Methods in image processing. Los Angeles; IEEE Computer Society, 1977.*

Ref. 3.5 *R. Kirsch. "Computer Determination of the Constituent Structure in Biological Images". Compt. Biomed. Res. 4 no. 3 (1971): 315-328.*

Ref. 3.6 *G. S. Robinson. "Edge Detection by Compass Gradient Masks." Computer Graphs and Image Processing 6 (1977): 492-501.*

Ref. 3.7 *W. Frei and C. C. Chen. "Fast Boundary Detection: A Generalization and a new algorithm ." IEEE Trans. Computer 26, no. 2 (October 1977): 988-998*

Ref. 5.1 *Anil K. Jain "Fundamentals of digital image processings", Chapter 1, Prentice-Hall Imternational, 1989.*

Ref. 5.2 *R. M. Haralick. "Statistical and Structural Approaches to Texture." Proc. IEEE 67 (May 1979) : 786-809.*

Ref. 5.3 *Anil K. Jain "Fundamentals of digital image processings", Chapter 9, Section 11, P395-P397, Prentice-Hall Imternational, 1989.*

Ref. 5.4 *B. S. Lipkin and A. Rosenfeld (eds). Picture Processing and Psychopictorics. New York: Academic Press, 1970.*

Ref. 5.5 *T. Pavlidis. Structural Patern Recognition. New York: Springer-Verlag, 1977.*

Ref. 5.6 *Anil K. Jain "Fundamentals of digital image processings", Chapter 9 Prentice-Hall Imternational, 1989.*

Ref. 5.7 *KALIVAS AND ALEXANDER A. SAWCHUK "A region matching motion estimation algorithm" Image Understanding Vol 54, No. 2, September, p. 309-324, 1991*

Ref. 5.8 *C. R. Brice and C. L. Fennema. "Scene Analysis Using Regions," Computer methods in Image Analysis. Los Angeles: IEEE Computer Society, 1977.*

Ref. 6.1 *Chapter 4 of this thesis.*

Ref. 6.2 *S. Lim. "Image Enhancement." In Digital Image Processing Techniques (M.P. Ekstrom. ed.) Chapter 1, pp. New York; Academic Press, 1984*

Ref. 6.3 *T. S. Huang (ed.). Two-dDimensional Digital Signal Processing I and II. Vol 42-43. Berlin : Springer Verlag, 1981*

Ref. 6.4 *Fernand S. Cohen and Zhigang Fan. Maximum Likelihood Unsupervised Textured Image Segmentation. CVGIP: Graphical Models and Image Processing Vol.54, No.3, May. pp. 239~251, 1992*

Ref. 6.5 *D. B. Cooper et al., Stochastic boundary estimation and object recognition, Comput. Graphics Image Process. Apr., 1980, 236~355*

Ref. 6.6 *C. Therrin, An Estimation-theoretic approach to terrain image segmentation, Comput. Graphics Image Process. 22, 1983.*

Ref. 6.7 *T. Pavlidis, Structure Pattern Recognition, Springer-Verlag, New York, 1977.*

Ref. 6.8 *A. Hanson et al., Experiments in schema-driven interpretation of a natural scene, in Digital Image Processing and Analysis (J. Simon, Ed.), Reidel, Dordrecht, The Netherlands, 1981.*

Ref. 7.1 *Gerhard Roth and Martin D. Levine, "Exctracting Geometric Primitives", Image Understanding Vol 58, No. 1, July, pp. 1-22, 1993*

Ref. 7.2 *James N. Huddleston and Jezekiel Ben-Arie "Grouping Edgels into Structural Entities Using Circular Symmetry, the Distributed Hough Transform, and Probabilistic Non-accidentalness" Image Understanding Vol 57, No. 2, March, pp. 227-242, 1993*

Ref. 7.3 *Todd R. Reed and J. M. Hans Du Buf, "A review of Recent Texture Segmentation and Feature Extraction Techniques", Image Understanding Vol 57, No. 3, May, pp. 395-372, 1993*

Ref. 7.4 *Chapter 4 of this thesis.*