

# **A RELATIONAL IMAGE DATABASE FOR SATELLITE IMAGERY**

A dissertation submitted in partial fulfilment of the requirements for the  
**M. Tech. (Computer Science)** degree of the Indian Statistical Institute

By

**K. Ravichandran**

Under The Supervision of

**Dr. Swapan Kumar Parui**

Computer Vision and Pattern Recognition Unit.

&

**Mr. Amitava Dutta**

Computer and Statistical Service Centre.

**INDIAN STATISTICAL INSTITUTE**

203, Barrackpore Trunk Road,

Calcutta-700035

July 23, 1996

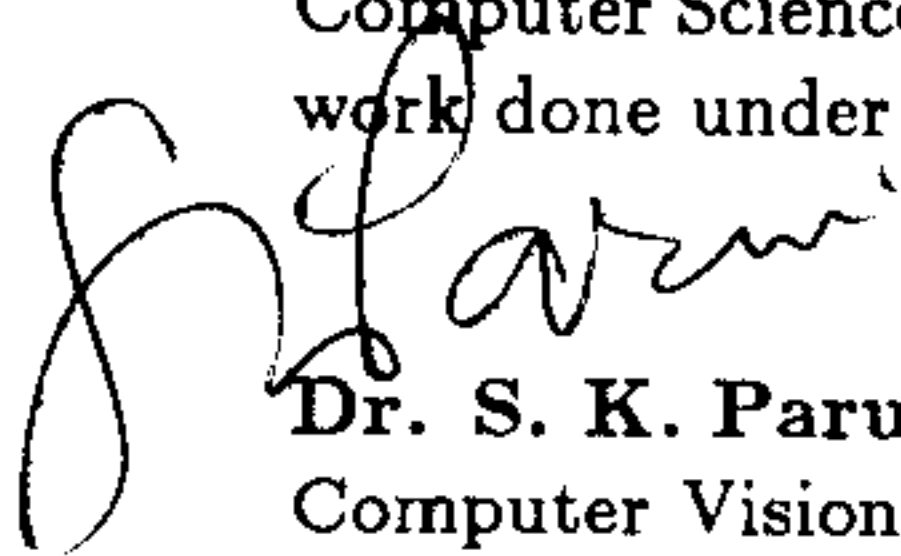


**Indian Statistical Institute**

203, B.T. Road,  
Calcutta- 700 035.

*Certificate of Approval*

This is to certify that the thesis titled **A RELATIONAL IMAGE DATABASE FOR SATELLITE IMAGERY** submitted by **K. Ravichandran.**, towards partial fulfilment of the requirements for the degree of M. Tech. in Computer Science at the Indian Statistical Institute, Calcutta, embodies the work done under my supervision.



**Dr. S. K. Parui.**  
Computer Vision and  
Pattern Recognition Unit.  
Indian Statistical Institute,  
Calcutta-700 035.



**Mr. Amitava Dutta**  
Computer and Statistical  
Service Centre.  
Indian Statistical Institute,  
Calcutta-700 035.

## Acknowledgements

At the outset I would like to express my profound gratitude to my guide Dr. S. K. Parui for his valuable suggestions during the project. Without his help and inspiration the project would never have been a success. I would like to thank Mr. Amitava Dutta for the valuable tips he gave for the project.

I would also like to thank Dr. B. B. Chaudhuri, Head, CVPR unit, for providing me with the computer resources needed for the project. I also wish to thank Mr. Umesh, Mr. Tamal, Mr. Umapada Pal and other members of the CVPR unit for their timely help and co-operation during the tenure of my project. My thanks goes to Mr. Anirban and Mr. Ujjwal for helping me to print the images. My thanks also goes to the staff of the CSSC for their help by way of providing the necessary resources.

Finally I would like to thank my classmates for making my stay in ISI memorable.

July 23,1996



(K. Ravichandran.)

# Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>2</b>
1.1	Requirements of Image Information Systems . . . . .	3
1.2	Goals of Image Information Systems . . . . .	4
<b>2</b>	<b>Image definitions</b>	<b>5</b>
<b>3</b>	<b>Image database system</b>	<b>8</b>
3.1	Image analysis and Pattern recognition . . . . .	8
3.2	Image structuring and understanding . . . . .	8
3.3	Spatial reasoning to aid in information retrieval . . . . .	9
3.4	Query Language and user interface . . . . .	9
3.5	Image information retrieval . . . . .	9
<b>4</b>	<b>Problem description and implementation</b>	<b>10</b>
4.1	Overview of Oracle (with regard to Images) . . . . .	10
<b>5</b>	<b>Image manipulation techniques and algorithms</b>	<b>13</b>
5.1	Histogram generation . . . . .	13
5.2	Histogram equalization . . . . .	14
5.3	Edge detection . . . . .	16
5.4	Image segmentation . . . . .	18
<b>6</b>	<b>Techniques for design of image database</b>	<b>19</b>
6.1	Image encoding in Image databases . . . . .	19
6.2	Indexing techniques . . . . .	24
6.3	Image matching by matching of 2D strings . . . . .	24
6.3.1	Representing of Symbolic Images by 2D strings . . . . .	24
6.3.2	Matching strategy . . . . .	25
<b>7</b>	<b>Conclusions</b>	<b>27</b>

# Chapter 1

## INTRODUCTION

One way of communication between men and machines is by means of images. Until recent times very little attention was paid to management of non-numeric information such as digital images, especially due to problems of storage. Recent advances in storage methodologies have made the handling of complex images efficient. Image information systems are needed as subsystems for advanced information systems in many application areas and as stand-alone systems depending on the practical application. The broad areas in which these are applied are :

*Fifth Generation computer systems* : These are knowledge information systems that can overcome technical restrictions inherent in conventional computers. Machine intelligence is vastly improved in these systems and they handle intelligent user interface functions like understanding speech, image, natural language and so on. For many of these an intelligent user interface capable of image communication and image understanding is required.

*Office information systems* : Ability to handle documents and forms is required for almost office information systems. The fields in these include signatures, photographs, special symbols and so on, which are both text and graphic data often needing to be created, encoded, edited, stored, retrieved and transmitted. Since these are generally distributed systems designed around LANS, multimedia communication is emphasized in such systems.

*Computer Aided Design* : CAD is an increasingly important image database application area. CAD/CAM database design is closely related to image database design.

*Image understanding systems* : These systems use image database as a subsystem.

A combination of image understanding, knowledge base and image information handling is required to design such a sophisticated system.

*Computer integrated manufacturing* : Image database is used as a subsystem for computer vision in Robotics and CAM. For instance, in the inspection of VLSI chips, thousands of mask patterns are required to be stored in a pattern database, which is an image database and faults are checked with respect to the reference patterns.

*Satellite imagery information system* : Everyday new remotely sensed data arrive from satellites and produce a huge number of large images. Thus an image database system is of great importance in this field. Moreover geographic information systems or cartographic information systems are very much in vogue.

*Medical information systems* : The concept of medical picture archiving and communication systems require design of sophisticated image information system with user interface and ability for image communication. Medical image information systems are of great use for automatic diagnosis of diseases and for training of pathologists.

## 1.1 Requirements of Image Information Systems

The characteristic features of Image Information Systems are :

1. Image Input : It involves the capturing and digitizing an image through a camera.
2. Image editing : It deals with changing the contents of a digitized image, or creating a new image from an old one.
3. Image processing : This stage comprises of image enhancement, edge detection, texture analysis, segmentation, and other statistical techniques of image transformation.
4. Image storage : It involves formatting, encoding, decoding, data structuring, and indexing for storage in a medium.
5. Image retrieval : The retrieval of an image from a image database by indexing or by query language or by similarity measures.

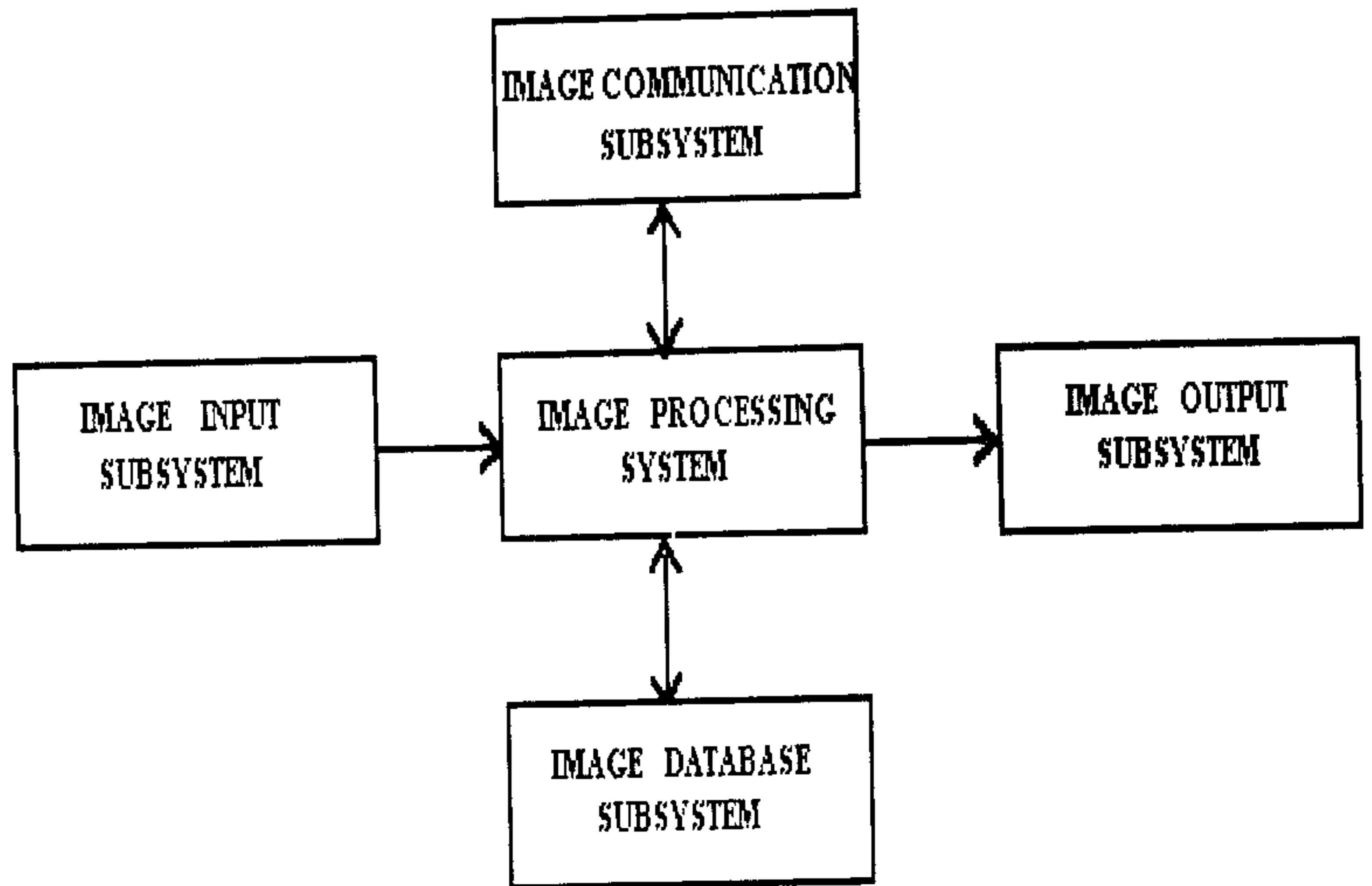


FIGURE 1: COMPONENTS OF AN IMAGE INFORMATION SYSTEM

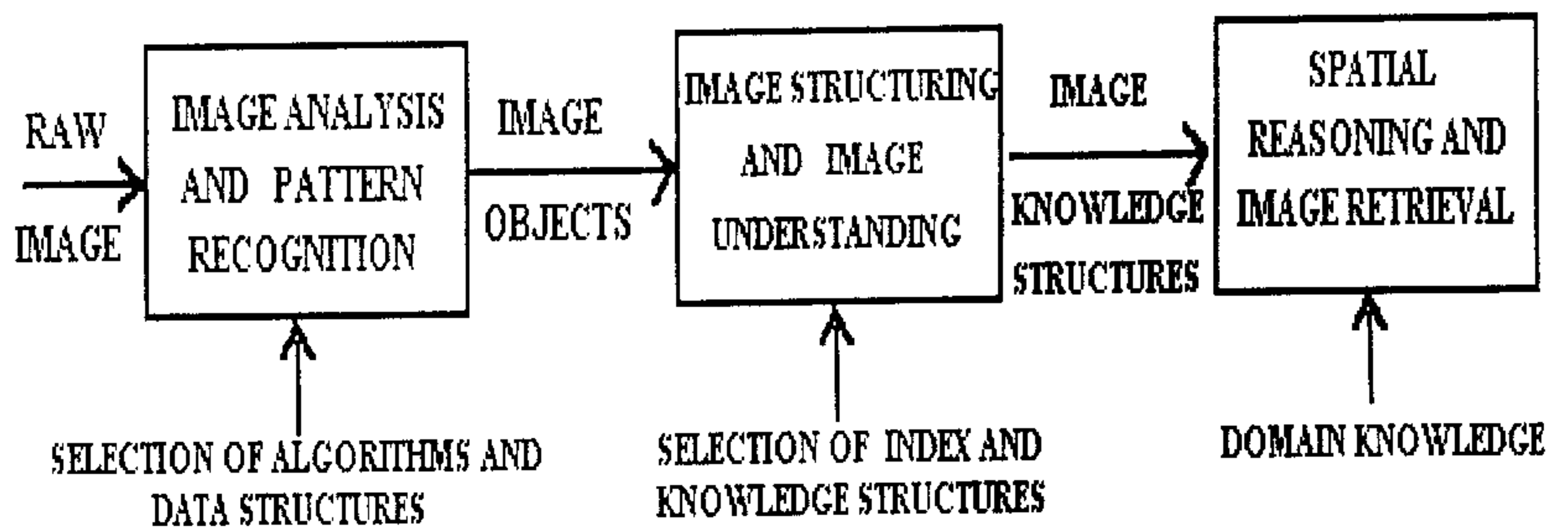


FIGURE 2 : STAGES IN KNOWLEDGE BASED IMAGE PROCESSING

6. Image output : It comprises of displaying a image and obtaining hard copies of it.
7. Image communication : This deals with the transmission of a image to a remote location through computers.

## **1.2 Goals of Image Information Systems**

The main goals may be summed up as below :

1. A large set of images are systematically collected and are available to a large number of users.
2. A database for retrieval of secondary information (eg. date of acquisition, altitude, longitude etc.,) of the above images.
3. A management system for imagery and map data for spatial oriented processing.

An image database is a system in which large amounts of image data and related information are integratedly stored.

The present work is an attempt to develop an Image database for Satellite images. This involves the following:

- (a) *Storing the images in the database (On an Oracle platform)*
- (b) *Performing image processing operations on the images*
- (c) *Querying strategies to retrieve images from the database*



# Chapter 2

## Image definitions

**Image function :** A image function  $f(x, y)$  is a real valued of two variables having values that are non-negative and bounded.

ie.  $0 \leq f(x, y) \leq L - 1 \quad \forall(x, y)$  assuming  $f$  is analytically well behaved.

The resultant digital image function  $f$  can be represented as an array of image elements/pixels which is a mapping from

$$\{ 0, 1, \dots, M - 1 \} \times \{ 0, 1, \dots, N - 1 \} \rightarrow \{ 0, 1, \dots, L - 1 \}$$

The two integers  $M$  and  $N$  represent the size of the digital picture.

The set  $\{ 0, 1, \dots, L - 1 \}$  is called the gray level set and  $L$  is the number of distinct gray levels. Each point  $(x, y)$  in the rectangular grid has pixel value  $f(x, y)$ .

A multispectral image function is represented by

$$f_f(x, y) = \{ f_1(x, y), f_2(x, y), \dots, f_n(x, y) \}$$

where  $f_1, f_2, \dots, f_n$  are image functions.

In color images with three basic colors (R,G,B) is represented by a multispectral image function.

$$f_f(x, y) = \{ f_R(x, y), f_G(x, y), f_B(x, y) \}$$

where each pixel  $f_f(x, y)$  is represented by  $(r, g, b)$  where  $r, g, b$  are pixel values of three pixel functions  $f_R, f_G, f_B$ .

**Image operations:**

To perform image operations consider the neighbour set NS which is a set of points regarded as neighbours of a point. The four point neighbour set NS4 is given by

$$NS4(x, y) = \{ (x - 1, y), (x + 1, y), (x, y + 1), (x, y - 1) \} \quad (2.1)$$

Each pixel is at unit distance from  $(x, y)$  and some of the neighbours may lie outside the digital image if  $(x, y)$  is on the border of the image. The four diagonal neighbours have co-ordinates

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

These points together with the set NS4 is called the eight point neighbour set NS8. Even in this case some points may fall outside the image if  $(x, y)$  is on the border of the image.

A image operation is an operation which when applied to a image transforms it into another image. Let F denote a family of digital images, then a image operation T is a mapping from F into F, or

$$T : F \rightarrow F$$

If the original image is  $f_1$  the transformed image  $f_2$  is

$$f_2 = T(f_1)$$

A image operation is called a local operation if  $f_2(x, y)$  is only dependent on the value of  $f_1(x, y)$  for  $(x, y)$  in NS(x,y).

If  $NS(x, y) = \{(x, y)\}$  and  $f_2(x, y) = T(f_1(x, y))$  then T is a point operation, which is a special case of a local operation.

Linear local operations are expressed by

$$f_2(x, y) = \sum_{(i, j) \in NS(i, j)} f_1(i, j) h(i, j)$$

The local h operator is sometimes called a window operator.

*Global operations:* These are useful in performing non-local image transformations such as geometric operations like image rotation. A point  $(x, y)$  in the

ideal image  $f$  is mapped onto a point  $(x', y')$  in the distorted image  $g$  where

$$\begin{aligned}x' &= T_1(x, y) \\y' &= T_2(x, y) \\g(x', y') &= g(T_1(x, y), T_2(x, y)) = f(x, y)\end{aligned}$$

Recovering the ideal image from  $g$  is done using the inverse operation

$$\begin{aligned}x &= T_1^{-1}(x', y') \\y &= T_2^{-1}(x', y')\end{aligned}$$

# Chapter 3

## Image database system

We shall concentrate on the Image Processing and Image database system which constitutes the heart of the Image information system. A traditional image processing system primarily performs the tasks of image analysis, image enhancement and pattern recognition [1]. Within this context an image database system should perform the following functions which are explained below.

### 3.1 Image analysis and Pattern recognition

The raw image is analysed and the image objects recognised. Extensive techniques are used for image enhancement, normalization, segmentation and other statistical operations. The end result is a collection of recognised image objects. These image objects are encoded in some data structures such as quad trees, run-length code, polygonal contour code, oct-trees etc. The image objects are those with attributes like co-ordinates, geometric properties etc. The input to this stage includes selection of image processing algorithms and selection of data structures.

### 3.2 Image structuring and understanding

The image objects are converted into image knowledge structures so that spatial reasoning and image information retrieval can be supported. The structure depends on the application domain knowledge and the features indexed. Other descriptors include shape descriptors, color descriptors and spatial relations such as hypergraphs or 2-D strings. The indexes can be used to access various data structures, or embedded into hierarchical image knowledge structures. Alterna-

tively, one may use directed graphs of spatial relations, semantic networks etc. Therefore the input to this stage includes the selection of knowledge structures.

### **3.3 Spatial reasoning to aid in information retrieval**

To solve specific problem in a domain, a knowledge base is needed. It is necessary to perform various transformations upon the image structure, so that the desired image knowledge can be easily accessed, visualized and/or manipulated resulting in a set of retrieved images, image indexes etc.

Conceptually these stages are regarded as generalized transformations to transform a raw image into an image data structure and then into image knowledge structure and finally into user-specific knowledge structure.

### **3.4 Query Language and user interface**

Many query languages developed for image information system are command languages or commands plus expressions following an SQL like syntax. One of the methods is the Query by Image example. Alternatively, in Query by example queries are specified using tables. An image query can be expressed directly as an image which is converted into 2-D strings for matching against an iconic index of the image database. Similarity retrieval supports query by image content in 2-D and even 3-D images ie. images having similar features are retrieved.

### **3.5 Image information retrieval**

In image information retrieval we can retrieve images satisfying a certain image query. One method is the quad tree approach which will be given later. An image query can also be represented by a 2D string. The 2D string subsequence matching is used for image information retrieval. Using this we can construct iconic indexes for images. This technique is also discussed at the end.

# Chapter 4

## Problem description and implementation

### 4.1 Overview of Oracle (with regard to Images)

Two types of images can be incorporated into an Oracle Forms application namely:

1. Boilerplate images.
2. Image items.

**Boilerplate images:** These are static images (either vector or bitmap) that can be imported from the file system database to use as graphical elements in the form, such as company logos and maps.

**Image items:** Image items are special types of interface control that store and display either vector/bitmap images. Like other items that store values, image items can be either base table items (items that relate directly to database columns) or control items. The definition of the image item is stored as part of the form module but no image file is actually associated with an image item until the item is populated at runtime.

At runtime it is possible to navigate to and select image items but it is not possible to navigate or modify boilerplate items.

Images and drawings are always stored in the database in Oracle format. During runtime the Oracle form inserts the image in the corresponding LONG RAW column in Oracle format to populate the image item associated with the TIFF file.

Many fields were identified for storage into the database in the case of Satellite imagery. Three tables were constructed for this, one for the sensor characteristics, another for the image characteristics and the final one for Satellite images.

**Table Sensor**

Field	Data type	No.of characters/digits
Sensor type	Character	10
Image number	Number	4
Year image was taken	Number	4
Resolution	Number	8
Spectral band type	Character	10
Dynamic range	Number	10
Bits/scene	Number	12

The second table discusses about the characteristics of the image.

**Table Image - char**

Field	Data type	No. of characters/digits
Image number	Number	4
Image description	Character	40
Image of the scene	Long raw	1 gigabyte
Identification	Character	15

The tables were constructed keeping in mind the queries that can be designed for the case of *Satellite* images. The image processing modules facilitate the answering of these queries and retrieval of the processed images from the database.

The other table signifies the satellite image characteristics namely:

Field	Data Type	No. of characters/digits
Latitude of first point	Number	5
Longitude of first point	Number	5
Latitude of second point	Number	5
Longitude of second point	Number	5
Landuse	Character	15
Altimetry	Number	6
Landslides in %	Number	3
Population density/sq.km	Number	6
Slope of ground	Number	2
Image number	Number	4

Various image processing modules were implemented and incorporated into the database. These are classified as under:

*Histogram evaluation and Histogram equalization:* This involves displaying the frequency of occurrence of each gray level value in the image.

*Edge detection:* We can also perform edge detection on the image by using the various operators namely Sobel's, Prewitt etc.

**Geometric modification modules:**

*Magnify:* Image size is expanded using this process.

*Reduce:* It involves reducing the image size.

*Rotate:* We rotates the image through a certain angle.

*Combine:* Geometrically combining two images and removing overlap.

*Intensity:* Here we apply intensity modification on the image. (uniform)

**Enhancement modules:**

*Filter:* Perform Fourier transform and filtering the image.

*Noise removal:* We eliminate noise from the image data.

*Laplace:* Here we compute the Laplacian of the image.



# Chapter 5

## Image manipulation techniques and algorithms

The major types of gray scale image processing algorithms which have been implemented are filtering, image enhancement, and segmentation. Most methods used for performing such processing use statistics computed on images such as histogram of distribution of gray levels and co-occurrence matrix of pairs of gray levels at pixel pairs, either directly or indirectly.

### 5.1 Histogram generation

Consider an image function having  $M \times M$  pixels with a pixel  $P$  (specified by its location) has a brightness level, or color  $Z$ . For monochrome image we assume that  $Z$  varies between 0 (very dark) and some value  $L > 0$  (very bright). Let  $f(P)$  denote the brightness levels at pixel  $P$  and  $H(Z)$  denote the histogram of the image.

**ALGORITHM** used to evaluate  $H(Z)$ . (**HISTOGRAM EVALUATION**).

*Notation:* -

$f(P)$  is the value of pixel  $P$  with range  $[0, L]$ .  $H$  is the histogram array.

Step 1. Initialize the array  $H(Z)$  ( $0 \leq Z \leq L$ ) to zero.

Step 2. For all pixels  $P$  of the image do:

Begin.

Step 3. Increment  $H(f(P))$  by 1.

End.

Step 4. End of the algorithm.

The histogram can be used for image enhancement or encoding. If  $H(Z)$  is

zero for many values of  $Z$  then it implies that the available levels of quantization are not used efficiently. By reassigning we can increase the dynamic range of the image and ensure that the histogram is as flat as possible. So we use histogram equalization for this process.

## 5.2 Histogram equalization

Let  $A$  be the image area and  $N$  be the number of available brightness levels for a perfectly flat histogram. Then there should be  $A/N$  pixels at each level. If the brightness value at some level  $Z$  is  $k$  times the average, then that level must be mapped into  $k$  different levels from say  $Z_1$  to  $Z_k$  [6] [7]. The rules for a one-to-many mapping are as follows:

Rule 1. Always map  $Z$  onto the midlevel  $(Z_1 + Z_k)/2$ . (This does not result in a flat histogram, but one where brightness levels are spaced apart).

Rule 2. Assign at random one of the levels in the interval  $[Z_1, Z_k]$ . (This can result in loss of contrast if the original histogram had two distinct peaks that were far apart).

Rule 3. Examine the neighbourhood of the pixel and assign to a level from  $[Z_1, Z_k]$  which is closest to the neighbourhood average. (This can cause edge smearing, and requires considerably more computation than the other two rules).

The first rule is purely heuristic. The motivation for the second rule is to reduce any systematic error. The third rule attempts to enforce some coherence among pixel levels. It assumes that the closer two pixels are on the plane, the more likely it is that they have similar values.

This kind of processing cannot be applied indiscriminately because equalization can often cause deterioration in the appearance of the image.

## ALGORITHM 2. (HISTOGRAM EQUALIZATION).

*Notation:* H is the histogram array.  $H_{int}$  is an integral of the histogram. Z denotes the old levels and R the new ones. Each Z is mapped onto an interval  $[\text{left}(Z), \text{right}(Z)]$ .

Step 0. Read an image, evaluate its histogram and store it in the array H. Let  $H_{avg}$  be its average value.

Step 1. Set  $R = 0$  and  $H_{int} = 0$ .

Step 2. For  $Z = 0$  to L do

    Begin.

Step 3.        Set  $\text{left}(Z) = R$  and add  $H(Z)$  to  $H_{int}$ .

Step 4.        While  $H_{int}$  is greater than  $H_{avg}$  do:

            Begin.

Step 5.                Sub  $H_{avg}$  from  $H_{int}$  and  
                        increment R.

            End.

Step 6.        Set  $\text{right}(Z) = R$  and define the value of  $\text{new}(Z)$  according to the rule used. For rule 1 set  $\text{new}(Z)$  to the average of  $\text{left}(Z)$  and  $\text{right}(Z)$ . For rule 2 set  $\text{new}(Z)$  to the new difference  $\text{right}(Z) - \text{left}(Z)$ . (For rule 3  $\text{new}(Z)$  is left undefined).

    End.

Step 7. For all pixels of the image do:

    Begin.

Step 8.        If  $\text{left}(f(P))$  equals  $\text{right}(f(P))$   
                    then set the new value of the pixel P  
                    to the  $\text{left}(f(P))$

Step 9.        else

            if rule 1 is used set the value of  $\text{new}(f(P))$   
            if rule 2 is used choose a point at random  
            from the interval  $[0, \text{new}(f(P))]$ , add its  
            value to  $\text{left}(f(P))$  and use the result for  
            the new value of P.

            if rule 3 is used, compute the average of  
            the neighbourhood of P. If it exceeds  $\text{right}(Z)$   
            use  $\text{right}(Z)$  as the new value. If it is below  
             $\text{left}(Z)$  then use  $\text{left}(Z)$  as the new value.

            Otherwise use the average as the new value.

    End.

The above algorithm implements histogram equalization for any of the rules given. Step 0 prepares the histogram according to the previous algorithm. Steps 1 to 7 do the equalization by mapping the old brightness level onto the new levels. Steps 8 to 10 compute the new image.

### 5.3 Edge detection

Edges are defined as amplitude discontinuities between different regions of the image. The edge detecting system makes use of 3x3 compass gradient masks which is well suited for digital implementation [9]. 2D masks operators are used for filtering and enhancement of images. 2D differentiations can be performed by convolving the original image with the compass gradient masks as shown in the figure(3). The compass name indicates slope direction of maximum response. eg. North gradient mask produces maximum output for vertical luminance changes. ie. for horizontal edges. The direction of horizontal edge could be from left to right or from right to left. The direction of edges corresponding to the eight compass gradient masks are shown in figure(4). The numbers 0, ..., 7 indicate the eight principal directions in the 3x3 grid.

The application of the first four masks to the 3x3 grid surrounding a picture element gives the gradient magnitude and direction. The gradient picture is obtained by taking the maximum gradient magnitude at each point. The mask which yields the maximum gradient determines the direction of the edge. The edge map given is a 2D array of numbers ranging from 0 to 7. The edge map is used to determine the local connectivity. (see fig(4)) If the direction at the centre of the 3x3 grid is  $k$  ( $k = 0, \dots, 7$ ) and if the direction preceding and succeeding the edge vectors are  $k-1, k$ , or  $k+1 \pmod{8}$  for any of the 8 compass directions then the edges are connected. The threshold map is used to determine whether the gradient value is large enough to accept or reject the presence the edge point. The presence of an edge is determined by examining the gradient values, edge direction map and threshold map simultaneously. If the edge vectors in a 3x3 grid surrounding a point satisfy the local connectivity condition depicted in the figure (4) and if they are above the threshold set by the threshold map then it is determined that there exists an edge point.

Masks for Edge detection

Direction of edge	Direction of gradient	KIRSK	PREWITT	SOBEL
0	N	5 5 5 -3 0 -3 -3 -3 -3	1 1 1 1 -2 1 -1 -1 -1	1 2 1 0 0 0 -1 -2 -1
1	NW	-3 5 5 -3 0 5 -3 -3 -3	1 1 1 1 -2 -1 1 -1 -1	2 1 0 1 0 -1 0 -2 -2
2	W	-3 -3 5 -3 0 5 -3 -3 5	1 1 -1 1 -2 -1 1 1 -1	1 0 -1 2 0 -2 1 0 -1
3	SW	-3 -3 -3 -3 0 5 -3 5 5	1 -1 -1 1 -2 -1 1 1 1	0 -1 -2 1 0 -1 2 1 0
4	S	-3 -3 -3 -3 0 -3 5 5 5	-1 -1 -1 1 -2 -1 1 1 1	-1 -2 -1 0 0 0 1 2 1
5	SE	-3 -3 -3 5 0 -3 5 5 -3	-1 -1 1 -1 -2 1 1 1 1	-2 -1 0 -1 0 1 0 1 2
6	E	5 -3 -3 5 0 -3 5 -3 -3	-1 1 1 -1 -2 1 -1 1 1	-1 0 1 -2 0 2 -1 0 1
7	NE	5 5 -3 5 0 -3 -3 -3 -3	1 1 1 -1 -2 -1 -1 -1 1	0 1 2 -1 0 1 -2 -1 0

Figure 3

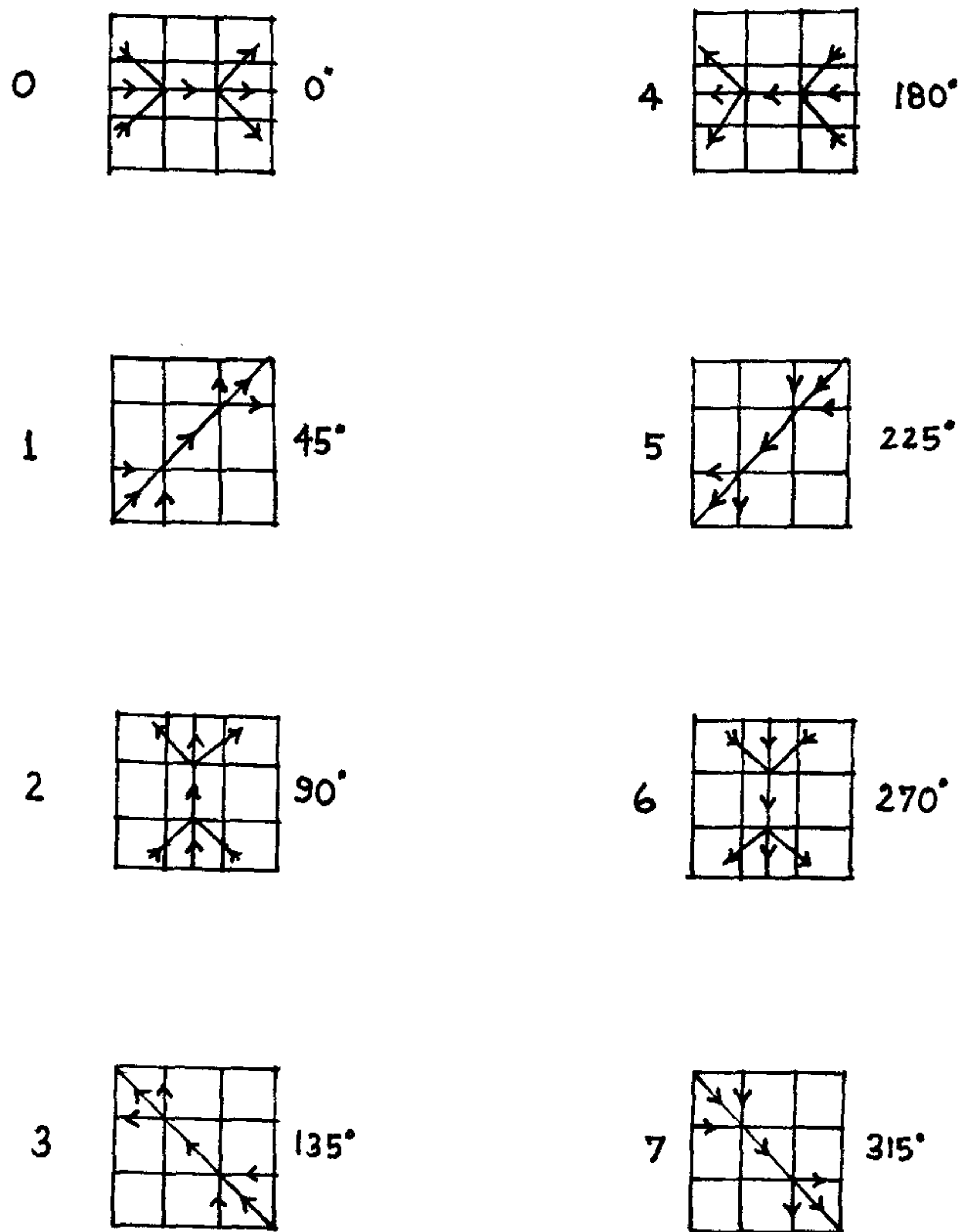
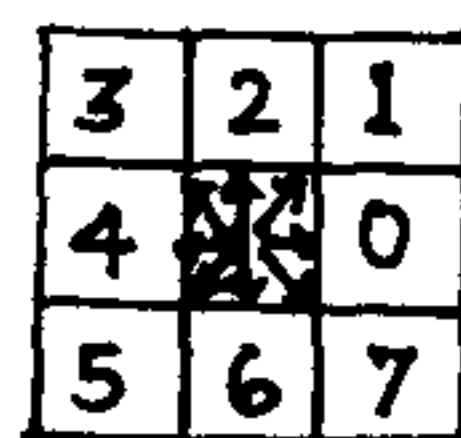


FIG 4. LOCAL CONNECTIVITY OF EDGES ON 3x3 GRID



8 PRINCIPAL DIRECTIONS ON 3x3 GRID

## 5.4 Image segmentation

**ALGORITHM** : A segmentation algorithm for use with image containing zeroes and a value.

Step 1. Given an image with  $m$  rows and  $n$  columns

$gz(i,j)$  for  $i=1, m$   $j=1, n$ .

$gz(i,j)$  = value for object.

$gz(i,j)$  = 0 for background.

Step 2. Set  $glabel = 2$  this is the label value.

Step 3. For( $i = 0; i < m; i++$ )

scan the  $i$ th row

For( $j = 0; j < n; j++$ )

check the  $j$ th element

stackempty = true

if ( $gz(i,j) == value$ )

    checkneighbour( $gz(i,j), glabel$ )

while (stackempty = false) do

    pop element ( $i,j$ ) off stack

    checkneighbour( $gz(i,j), glabel$ )

endwhile

$glabel = glabel + 1$

end of checking  $j$ th element

end of scanning  $i$ th row

Step 4. End.

Procedure checkneighbour( $gz(r,e), glabel$ )

Step 1.  $g(r,e) = glabel$

Step 2. for( $i = r - 1; i < r + 1; i++$ )

    for( $j = e - 1; j < e + 1; j++$ )

        if ( $gz(i,j) == value$ ) then

            push( $i,j$ ) onto stack

            stackempty = false

        endif

    end loop over  $j$

end loop over  $i$

Step 3. End procedure checkneighbour.

# Chapter 6

## Techniques for design of image database

### 6.1 Image encoding in Image databases

In Image database design we try to partition a large image into smaller pieces, sometimes called pages or tiles, so that the pieces can be stored economically and retrieved easily. The pieces of the image are often organised hierarchically into a tree structure.

Alternatively, we can use icons which is a image or a sketch referring to an object or a set of objects represented by a graphical sketch or a low-resolution image (called icon-image) on the display screen. Icons are associated with a hierarchical structure to serve as indexes for interactive information retrieval [1].

The image encoding problem in the context of Image database is that of partitioning of a large image into smaller pieces, the organisation of such pieces into a hierarchical structure and construction of icon-images to be associated with such a structure to facilitate image information storage and retrieval.

A data structure which is used to represent image data in a compact way is the Quadtree representation. A quadtree is based on the principle of recursive decomposition of the image. For a digital image in a 2D grid each decomposition produces four equal sized quadrants. A quadrant is labelled white if it consists of white pixels only, black if it consists of black pixels only, and gray otherwise. Further decomposition is carried on the gray blocks, black and white blocks remaining changed. The quad tree has its root representing the entire image, the leaves representing either black or white quadrants.

We assume that we start with an image that can be accessed row by row. To form a tree we read two such rows and then proceed to examine quadruples of



pixels in the order given in the figure(5). For each group of four pixels with indices  $2k, 2k + 1, \dots, 2^n + 2k, 2^n + 2k + 1 (k = 0, 1, \dots, 2^{n-1} - 1)$  let  $f_0, f_1, f_2$  and  $f_3$  be the corresponding gray levels. We create four new levels.

$$g_0 = \frac{1}{4} \sum_{i=0}^3 f_i \quad (6.1)$$

$$g_j = f_j - g_0 \quad j = 1, 2, 3 \quad (6.2)$$

Note that there is no loss in going from the  $f_j$ 's to  $g_j$ 's because we can recover  $f_j$ 's from  $g_j$ 's by the formula,

$$f_j = g_j + g_0 \quad j = 1, 2, 3 \quad (6.3)$$

$$f_0 = g_0 - \sum_{i=1}^3 g_i \quad (6.4)$$

To form the next level of the tree, we create a new image with a row whose elements are the  $g_0$ 's. At the same time we place in a separate array D the difference values  $g_1, g_2, g_3$ . Then we arrive at figure(6). When the next two rows are read in, the index of the first element will increase by  $2^{n+1}$  where  $2^n$  is the length of the row so that arrangement of figure(7) results. At the end we would have created a new  $2^{n-1} \times 2^{n-1}$  image plus an array D of  $3 \times 2^{2n-2}$  differences. This procedure can be repeated until we reach an image consisting of a single pixel. Then almost all the information will be contained in array D.

**ALGORITHM. Creating a level of a Quad tree.**

Procedure *TLEVEL*(*n,I,J,D*)

*Notation:* The Input is the image I of dimension  $2^n \times 2^n$ . Output is the image J of dimension  $2^{n-1} \times 2^{n-1}$  and array D contains the differences.

Step 1. For j from 0 to  $2^n$  by 2 do

**Begin.**

Step 2. Place the jth row of the image I into the array p1 and j+1 th row into the array p2.

Step 3. For i=0 to  $2^n$  by 2 do:

**Begin.**

Step 4. Set  $f_0$  equal to p1(i)

Set  $f_1$  to p1(i+1)

Set  $f_2$  to p2(i)

Set  $f_3$  to p2(i+1).

Step 5. Compute  $g_0, g_1, g_2, g_3$  by the given equations (6.1)

Step 6. Append  $g_0$  to the image J

Step 7. Append  $g_1, g_2, g_3$  to the array D

**End.**

**End.**

Step 8. End of the procedure.

**Algorithm : Creating a Quadtree.**

Step 0. Read the image  $I_n$  of size  $2^n \times 2^n$ .

Step 1. Set level = n.

Step 2. **While** (level is greater than 0) **do:**

**Begin.**

Step 3. Call *TLEVEL*(level,  $I_{level}$ ,  $I_{level-1}$ ,  $D_{level}$ )

Step 4. Decrement level by 1.

**End.**

Step 5. Write  $I_0, D_1, D_2, \dots, D_n$ .

Step 6. **End of the Algorithm.**

**Reconstruction of the image from the Quadtree.**

Procedure *FILL*(*i,xs,ys,xe,ye*)

Step 1. createwindow(xs,ys,xe,ye) /\* Define window on the screen \*/

Step 2. setbkcolor(i) /\* Set background color \*/

Step 3. erase /\* Erase window area to background intensity \*/

Step 4. **End of the procedure.**

Figure(5)

0	1	4	5	10	11	14	15	20	21	.....
2	3	6	7	12	13	16	17	22	23	.....
100	101	104	105	110	111	114	115	120	121	.....
102	103	106	107	112	113	116	117	122	123	.....

*Note : Order in which pixels are read from an array in order to form the quad tree. Labels are in octal assuming an  $64_{10} \times 64_{10}$  image.*

Figure(6)

$g_0$	$g_4$	$g_{10}$	$g_{14}$	$g_{20}$	$g_{24}$	.....
-------	-------	----------	----------	----------	----------	-------

$(g_1, g_2, g_3) (g_5, g_6, g_7) \dots\dots\dots (g_{75}, g_{76}, g_{77})$

*Note : Top row shows single row produced from 2 rows in original image and bottom row shows array of differences.*

Figure(7)

$g_0$	$g_4$	$g_{10}$	$g_{14}$	$g_{20}$	$g_{24}$	$g_{30}$	.....
$g_{100}$	$g_{104}$	$g_{110}$	$g_{114}$	$g_{120}$	$g_{124}$	$g_{130}$	.....

*Note : Elements of the first two rows of the new image.*

**Algorithm: Display the image.**

*Notation* : Input is the array D whose elements are differences from the local average except for the first element, which equals the average for the whole image.

- Step 1. Set I to the first element of array D and execute  $FILL(I, 0, 0, 2^n, 2^n)$ .
- Step 2. For i from 0 to n-1 do:  
    **Begin.**
- Step 3.                   Set  $l = 4^i$  and  $u = 2^{n-i}$
- Step 4.                   For j = 0 to l-1 do:  
                          **Begin.**
- Step 5.                   Read the next 3 elements of D namely  
                          g1,g2,g3.
- Step 6.                   Divide j by  $2^i$  and let q  
                          be the quotient and r be the  
                          remainder. Let  $y = 2qu$  and  $x = 2ru$ .
- Step 7.                   Read  $g_0$  from the image at  
                          location  $(x, y)$  and compute  $f_0, f_1, f_2,$   
                           $f_3$  from the EQNS(6.3,6.4)
- Step 8.                   Call  $FILL(f_0, x, y, x + u, y + u)$   
                          Call  $FILL(f_1, x + u, y, x + 2u, y + u)$   
                          Call  $FILL(f_2, x, y + u, x + u, y + 2u)$   
                          Call  $FILL(f_3, x + u, y + u, x + 2u, y + 2u)$
- End.**
- End.**
- Step 9. **End of the Algorithm.**

The problem of image database as stated earlier was the storage, manipulation, and retrieval of a images. For handling large number of images, some of which are complex, efficient and flexible retrieval mechanisms are required. The main idea is to represent image information by both logical pictures and physical pictures. A logical picture can be regarded as a model of the real image defined as the hierarchically structured collection of image objects. The logical picture can thus be stored as relational tables in the relational database and manipulated using the relational database manipulation language. Inquiries concerning the attributes of image objects are also handled by the RDBMS tools. Once the logical picture has been identified for retrieval, the corresponding physical picture can be generated on the output device by retrieving the physical picture from the Image storage device.

## 6.2 Indexing techniques

Indexing techniques comprises of three aspects namely index representation, index organisation and index extraction [1] [4]. Contrary to conventional database where key-word based indexing is prevalent, in image information system we often have to retrieve images by shape, texture, spatial relationships etc., ie. image features are used as indexes called as image indexes possessing the following characteristics:

1. Image indexes are approximately represented.
2. Image indexes do not have embedded order, ie. if a,b,c are three index values and  $a < b < c$ , it does not mean that image b is more similar to image a than image c is.

Image indexes are shape descriptors, signatures, and 2D strings.

### Types of Retrieval Strategies.

In image query translation system we classify image information retrieval into four categories, namely attribute, spatial/ structural, similarity, and complex retrieval.

*Attribute retrieval:* In this strategy of retrieval image objects are retrieved by logical attributes which are physically existing in the relational database.

*Spatial retrieval:* Here the image objects are retrieved by spatial relations and/or structural properties.

*Similarity retrieval:* In this method we retrieve image objects which are similar to a given image object using certain similarity measures.

*Complex retrieval:* This mode of retrieval is the combination of the preceding three types of retrieval commands.

## 6.3 Image matching by matching of 2D strings

### 6.3.1 Representing of Symbolic Images by 2D strings

Let  $V$  be a set of symbols called the vocabulary with each symbol representing a image object, a pixel and so on. Let  $A$  be the set  $\{=, <, :\}$  where  $\{=, <, :\}$  are three special symbols not in  $V$ , representing spatial relationship among image objects.

A 1D string over  $V$  is any string  $x_1, \dots, x_n$  where  $n \geq 0$  and  $x_i \in V \forall i = 1, \dots, n$ .

A 2D string over  $V$  written as  $(u,v)$  is defined as

$(x_1y_1x_2y_2 \dots y_{n-1}x_n, x_{p(1)}z_1x_{p(2)} \dots z_{n-1}x_{p(n)})$

where

$x_1, \dots, x_n$  is a 1D string over  $V$ .

$p: \{1, \dots, n\} \rightarrow \{1, \dots, n\}$  is a permutation over  $1, \dots, n$ .

$y_1, \dots, y_{n-1}$  is a 1D string over  $A$ .

$z_1, \dots, z_{n-1}$  is a 1D string over  $A$ .

The symbol ' $<$ ' denotes left-right spatial relation in string  $u$  and below above spatial relation in string  $v$ . The symbol '=' denotes spatial 'at the same location as'. The ':' denotes the relation 'in the same set as'. The 2D string representation can be seen as the symbolic projection of image  $f$  along the  $x$  and  $y$  directions.

A symbolic image  $f$  is a mapping  $M \times M \rightarrow W$  where  $M = \{1, 2, \dots, m\}$  and  $W$  is the power set of  $V$  where the empty set denotes the null object, ie. the blank slots.

### 6.3.2 Matching strategy

2D string representation provides a simple approach to perform subpicture matching on 2D strings [3] [4]. The rank of each symbol in a string  $u$  which is defined as one plus the number of ' $<$ ' preceding this string in  $u$ , plays an important role in 2D string matching. We denote the rank of a symbol  $b$  by  $r(b)$ .

*Def:* A string  $u$  is contained in a string  $v$ , if  $u$  is a subsequence of a permutation string of  $v$ .

*Def:* A string  $u$  is a type- $i$  1D subsequence of string  $v$ , if

1.  $u$  is contained in  $v$  and
2. if  $a_1 w_1 b_1$  is a substring of  $u$ ,  $a_1$  matches  $a_2$  in  $v$ , and  $b_1$  matches  $b_2$  in  $v$ , then
  - (type 0)  $r(b_2) - r(a_2) \geq r(b_1) - r(a_1)$  or  $r(b_1) - r(a_1) = 0$
  - (type 1)  $r(b_2) - r(a_2) \geq r(b_1) - r(a_1)$  or  $r(b_2) - r(a_2) = r(b_1) - r(a_1) = 0$
  - (type 2)  $r(b_2) - r(a_2) = r(b_1) - r(a_1)$

Now we define the notion of type- $i$  ( $i = 0, 1, 2$ ) 2D subsequence as follows. Let  $(u, v)$  and  $(u', v')$  be 2D string representation of  $f$  and  $f'$  respectively.  $(u', v')$  is a type- $i$  2D subsequence of  $(u, v)$  if

1.  $u'$  is a type- $i$  1D subsequence of  $u$ .
2.  $v'$  is a type- $i$  1D subpicture of  $f$ .

We say  $f'$  is a type- $i$  subpicture of  $f$ .

### *Procedure for Image matching*

1. Convert  $(u,w)$  and  $(u',w')$  to  $(x,r,s,p)$  and  $(x',r',s',p')$  respectively. String  $x$  over  $A$  consists of symbols appearing in  $u$  string,  $r$  is the corresponding rank string,  $s$  is another rank string corresponding to string  $y$  which consists of symbols appearing in  $v$  string and  $p$  is the permutation function. The same is true for  $(u',w')$  and  $(x',r',s',p')$ .
2. Construct initial matching table  
 $MI(n) = \{j \mid \text{whenever } y(j) = x(p(j)) = x(p'(j)) = y'(n)\}$  for  $n = 1, 2, \dots, N$
3. Check to see if there exists a type- $i$  subsequence in  $(u,w)$  which matches  $(u',w')$ .

### **Visualization of symbolic pictures:**

Given a 2D string representation  $(u,v)$  we can reconstruct the symbolic picture it represents.

*Algorithm for reconstruction:*

*2Dpicture(f,m1,m2,u,v,n)*

**begin**

    /\* find out size of the picture \*/

Step 1.  $m1 = 1 + \text{number of } '<' \text{ in string } u;$

Step 2.  $m2 = 1 + \text{number of } '<' \text{ in string } v;$

    /\* find out x-rank and y-rank of each object \*/

Step 3. **for**  $i$  from 1 until  $n$

**begin**

Step 4.                    $x\text{-rank of } x_i = 1 + \text{number of } '<' \text{ preceding } x_i \text{ in } u;$

Step 5.                    $y\text{-rank of } x_{p(i)} = 1 + \text{number of } '<' \text{ preceding } x_{p(i)} \text{ in } v;$

**end**

    /\* Construct an  $m1$  by  $m2$  picture \*/

Step 6. **for**  $j$  from 1 until  $m1$

Step 7.                   **for**  $k$  from 1 until  $m2$

Step 8.                    $f(j,k) = \text{the set of all objects with } x\text{-rank } j \text{ and } y\text{-rank } k;$

**end**

# Chapter 7

## Conclusions

In this work attempt has been made to store, manipulate and retrieve images from the database. Though some satisfactory results have been obtained it can be upgraded so that complicated queries may be handled efficiently. We conclude by giving some aspects by which this can be done :

- Other image data structures like B-tree, K-D tree can be explored for representing image data and the best one considered for usage.
- Other types of image indexes such as shape descriptors, signatures etc. can be taken as the other choices for indexing.
- Facility for retrieval of images by texture, shape etc. can be provided.



# Bibliography

1. Shi-Kuo Chang and Arding Hsu, "Image information systems," IEEE transactions on Knowledge and data engineering, Vol 4, No.5, pp 431-442 Oct.1992.
2. H. Tamura and N. Yokoya, "Image database system, a survey", Pattern recognition, Vol 17, No 1,pp 29-43, 1984.
3. W.I. Groskya and Rajiv Mehrotra "Index based object recognition in Pictorial data management", Computer vision, Graphics and image proessing, Vol 52, No 3, pp 416-436, 1990.
4. S.K. Chang , Q.Y. Shi, and C.W. Yan, "Iconic indexing by 2D strings," IEEE transactions on Pattern and machine intellegence-9, No. 3, May 1987, pp 413-428.
5. S.K. Chang, C.W. Yan, T. Arndt and D. Dimitroff,"An intellegent image database system", IEEE trans. in Software engg., pp 681-688, May 1988.
6. Gonzales and P. Wintz, Digital Image processing, Addison Wesley, 2nd edition, Reading Massachussets, 1987.
7. Anil K. Jain, Fundementals of Digital image processing, Prentice hall of India, 1995.
8. B.B. Chaudhuri, and D. Dutta Majumdar, Two-tone image processing and recognition, Wiley Eastern, 1993.
9. G.S. Robinson, "Edge detection by Compass gradient masks," Computer Graphics and image processing, (6), pp 492-501, 1977.
10. G.Y. Tang, "A management system for an integrated database of pictures and alphanumeric data", Computer Graphics and image processing, Vol 16, No.5, pp 270-286, 1981.