CHAPTER 3
METHODOLOGY

Reliable authentication of persons is a growing demanding service in many fields, not only in police and legal environments but also in civilian applications, such as access control or financial transactions. It can also be understood from the literature survey that even though many recognition techniques have been devised, none of the techniques proposed are 100% safe and accurate. Thus, it is clear that advancements in the partial fingerprint domain are imminent.

In this research, an approach for automatic fingerprint recognition that considers the use of features other than minutiae are considered to answer the challenges of Partial Fingerprint Recognition (PFR). The proposed Automatic Partial Fingerprint Recognition System (APFS) consists of three major steps (Figure 3.1). They are

(i) Fingerprint Acquisition (Sensors and data storage components),
(ii) Identification and Recognition, and
(iii) Matching and Decision Process.

Figure 3.1: Partial Fingerprint Recognition System
A fingerprint sensor is used to collect the fingerprints, which are converted to a digital format and stored as templates. The template is used during the decision process, where a new fingerprint is compared with the template. After acquisition, an input partial fingerprint is presented and a search with template fingerprints is performed (one-to-many matching). For this purpose, a feature fusion technology is proposed. The feature fusion technology presented combines various non-minutiae features namely, LBP around pores (level 3 fingerprint feature) and SIFT are used. The anticipated result of search is a match or non-match. Then, a decision process uses this result to make a system level decision. The matching of the partial fingerprint against the full-sized database is performed in two ways. The first method uses the traditional score-level matching and the second method analyzes the use of machine learning classifier with both single and fused feature sets.

3.1. PROPOSED METHODOLOGY

To develop the proposed partial fingerprint recognition system, various techniques and algorithms are combined and the methodology consists of the following steps.

1. Preprocessing – Consists of fingerprint image enhancement techniques.
2. Segmentation – Consists of procedures and algorithms that separate the Region of Interest (ROI) from the background. In this study, the ROI is the fingerprint.
3. Feature Selection – Consists of techniques that can efficiently and accurately extract pore, LBP and SIFT features from both the partial and full fingerprints.
4. Matching – Consists of methods to obtain the similarity scores that are used during recognition.

Each of the above steps is dealt separately and the techniques that enhance the operation of each of the steps, for ultimate gain during recognition, are proposed. Each step is interrelated with one another and the output of one
step is taken as input by the next step. Each step is considered as a separate and independent phase in the study. The advantages obtained due to the enhancements made are analyzed in the final phase using various performance metrics and different fingerprint datasets. Thus, the study has five phases, each focused on one aspect of partial fingerprint recognition. In the proposed APFS framework, the preprocessing and segmentation are set as optional arguments, which can be switched off and on, according to the requirement. This facilitates the researcher to study the impact of having preprocessing and segmentation during recognition. The various techniques and methods used in the proposed partial fingerprint with the above five phases are presented in Figure 3.2.

Fingerprint images (both partial and full print) often are degraded in image quality and require extensive image processing to make them suitable for subsequent steps like segmentation and feature extraction. The images produced by fingerprint sensors have to be processed in such a way that the visual appearance of the fingerprint is converted to a form that is better suited for analysis and understanding. The techniques and methods used for this purpose are termed as ‘Enhancement or Preprocessing Techniques’ and are used to improve fingerprint appearance by improving image features or by decreasing ambiguity between different regions of the image (Chanda and Majumder, 2002; Pratt, 2007).

The quality of fingerprint images is assessed using several quality parameters like brightness or contrast sensitivity, color unevenness and noise. The quality parameters can be altered by various factors like environment, exposure, skin type, movement during acquisition, sensitivity of the sensors, etc., all of which can produce images that degrade the performance of other steps. This necessitates techniques that can improve these quality parameters.
NIST SD 30 database (1000 dpi)
DPLS Dataset (500 dpi)

**PHASE I**
Preprocessing

- Image Enhancement
  - Auto Brightness, Contrast and Color Level Adjustment
  - Noise Removal

**PHASE II**
Segmentation

- Enhanced Harris Corner
- Enhanced Susan Corner

**PHASE III**
Feature Extraction

- LBP around Pore Features
- SIFT Features

**PHASE IV**
Matching

- Score based
- Neural Network based
  - LBP - Pore Features
  - SIFT Features
  - LBP-Pore + SIFT Features

**PHASE V**
Performance Evaluation

- Preprocessing – PSNR, FOM, MSSI, Time
- Segmentation – Stability ratio, Anti-noise criterion ratio, Speed
- Feature Extraction – Detection Rate
- Matching – FAR, FRR, EER, Accuracy, Execution Time

Figure 3.2: Research Methodology
The first phase of the research work is focused on developing such enhancement techniques to improve the quality of fingerprint images.

This phase focuses on four main preprocessing tasks, namely,

(i) Auto Brightness Adjustment
(ii) Auto Contrast Adjustment
(iii) Auto Color Level Adjustment and
(iv) Image Denoising (Fingerprint images normally acquire impulse noise (Salt & Pepper, Random)).

The algorithm performs image enhancement in two steps. The first step performs a simultaneous operation to adjust brightness, contrast and color levels. In this procedure, the image is first converted to a YUV color space. The luminance is used for adjusting brightness and contrast, while the Q component is used for color level adjustment. In the second step, an enhanced Vector Median Filter (VMF) is used to remove the impulse noise. The enhanced version of VMF has the advantage of edge preservation and reduction of computation complexity.

The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze (Shapiro and Stockman, 2001). Segmentation is the process of partitioning an image into non-intersecting regions such that each region is homogeneous and the union of no two adjacent regions is homogeneous. In this study, it is used to separate the fingerprint from the rest of the image. The main aim of this phase is to identify the fingerprint area by splitting the fingerprint into two regions, namely, Foreground and Background. The foreground will have the Region Of Interest (ROI), that is, the fingerprint and the unwanted background is removed from further analysis. The advantage of this phase is that, since the unwanted regions are removed, the computations are performed only on the ROI, thus saving processing time and cost.
Two most frequently used approaches for segmentation are pixel-based and edge-based. Out of these two, edge-based techniques are more frequently used. Traditional edge-based algorithms efficiently identify edges, but work poorly with corners. So the present research work combines edge based segmentation technique with two corner-based segmentation algorithms, namely, Harris Corner and Susan Corner detection algorithms. The enhanced algorithms have the advantage of working efficiently both with edges and corners.

In pattern recognition and image processing, feature selection and extraction is a special technique used for selecting a subset of most relevant image characteristics that can best represent the whole image. Transforming the input digital image data into a set of features is called feature extraction. If the method of extraction is designed efficiently, then the subsequent recognition task can be performed with this reduced representation instead of the full sized input image.

The minutiae and pore features are the two frequently used features extracted from a fingerprint image. The accuracy of the recognition system is directly proportional to the number of features extracted. While considering partial fingerprints, it is a known fact that the number of features that can be extracted is small. To solve this problem and to increase the performance of the recognizer, the present study considers the following during feature extraction.

- Extraction of non-minutiae features like Local Binary Patterns (LBP) around pores and Scale Invariant Feature Transform (SIFT) descriptors for partial fingerprint recognition.
- Feature fusion to enhance accuracy and robustness of the recognizer.

Non-minutiae features are considered because the acquired partial fingerprint can often be rotated by any angle and can map to any region of the original full fingerprint. This makes the correct mapping of partial fingerprint to the correct region of the full fingerprint a challenging task. To answer this
challenge, this study uses two non-minutiae features, namely, LBP around pores and SIFT features.

Both the features have the common objective of improving the accuracy and robustness of the proposed recognizers. The LBP around pores offer multiple advantages like being rotation invariant, free from gray scale invariance, robust to noise and small skin distortions. Also since LBP around pores are rotation invariant, alignment is not required. Similarly, the SIFT feature apart from being reasonably invariant to noise and rotation, can also utilize the texture information both for extracting feature points and matching.

Further, to take advantage of both the feature sets, the study applies a feature-level fusion that combines LBP around pores and SIFT features to create a fused feature vector. The advantage of feature level fusion lies in two aspects: First, it can derive the most discriminatory information from original multiple feature sets involved in fusion; second, it enables eliminating redundant information resulting from the correlation between distinct feature sets and making the subsequent decision in real time possible. In short, feature fusion is capable of deriving and gaining the most effective and least-dimensional feature vector sets that benefit the final decision. Thus, the study uses three types of features, namely, LBP around pores, SIFT features and fusion of LBP around pores and SIFT features.

Two matching algorithms are used during the identification process of the partial fingerprint with the full fingerprint template. They are score based matching and neural network-based matching. Using the three feature vectors and two matching algorithms, six recognizers are designed. They are LBP around pores using Score based recognizer, SIFT using Score based recognizer, Fusion feature using Score based recognizer, LBP around pores using BPNN recognizer, SIFT using BPNN recognizer and Fusion feature using BPNN recognizer.
3.2. RESEARCH CONTRIBUTIONS

This research work aims to enhance each of the various steps of partial fingerprint recognition system. The various steps are preprocessing, segmentation, feature extraction and matching.

The proposed preprocessing model proposes a single-model algorithm that adjusts brightness, contrast, color-level and noise removal. The noise removal algorithm enhances the traditional Vector Median Filter (VMF) to remove impulsive noise from fingerprints. The traditional algorithm fails to distinguish thin lines and boundaries during noise removal and usually filters them out because it interprets these fine details as noise. Moreover, the algorithm suffers from high computation cost due to repeated distance calculation of similar values in the filtering window. The enhanced version solves both these problems by using a procedure to differentiate edges / boundaries from other regions and applying VMF only to other regions. The problem is solved by minimizing the repeated calculations involved. This is performed by using a procedure that calculates the minimum distance in a fast manner and applying VMF only to those pixels that are affected by impulse noise. The problems of more than one pixel having the same minimum distance and color distortion are solved by using a simple rule-based distance calculation.

The proposed segmentation algorithm aims to extract the fingerprint from background to reduce the processing time and cost. The proposed method combines edge-based and two-frequently used corner-based techniques, namely, Harris-Corner and Susan-Corner. The disadvantage of Harris and Susan is that the computing derivative is sensitive to noise and therefore generate “false” corners. Moreover, it has poor localization performance because it needs to smooth the derivatives for noise reduction. Both these problems are solved by first finding the edges of the image and then performing Harris or Susan only on the detected edges. This saves time and
avoids false detection and is not sensitive to noise. The present research work analyzes the performance of the various traditional edge detection algorithms (Canny, Sobel, Roberts, LoG and Prewitt) and selects the best. Experiments show that Canny is efficient and hence is used.

In the feature selection and extraction part of the proposed recognizer, techniques that eliminate the need for alignment during matching are proposed. For this purpose, LBP around pores and SIFT features are proposed for partial fingerprint recognition. Selecting LBP around pores is a novel contribution made in this study where Marker Controlled watershed Segmentation algorithm is used to extract the pores. A feature fusion technique is proposed to combine the advantages of both the selected features and to increase the efficiency of the proposed recognizer. During matching and decision making step, two methods are used. The first is a score based matching algorithm and the second is the use of machine learning algorithms that use the fused features. The methodology behind each contribution is discussed in the following sections.

This research investigating level 3 pore features and non-minutiae features for partial fingerprint recognition was sponsored in part by UGC (No.F. 34-103/2008 SR) as Major Research Project.

3.3. EDGE DETECTION ALGORITHMS

In the present study, edge detection plays an important role both during preprocessing and segmentation. Five popular traditional edge detection operators are examined for its applicability to detect edges efficiently from the fingerprint images. Selection of edge detection is based on their ability to detect edges in an accurate manner. The selected edge detectors are

- Sobel Operator
- Prewitt Operator
- Roberts Operator
- Laplacian of Gaussian (LoG) Operator
- Canny Operator
Edge detection process involves small kernels to convolve with an image to estimate the first-order directional derivatives of the image brightness distribution. Kernels are a pre-defined group of edge patterns to match each image segments of a fixed size. The edge value is calculated by forming a matrix centered on each pixel. If the value is larger than a given threshold, then the pixel is classified as an edge. All the gradient-based algorithms have kernel operators that calculate the edge strength in directions that are orthogonal to each other, commonly vertically and horizontally. The contributions of the both the components are combined to give the total value of the edge strength. The kernel used by different classical edge detectors are given below.

Edge detection algorithms are grouped into two categories, namely, Gradient operator and Laplacian operator (Harris and Stephens, 1988). Gradient operator detects edge pixel by obtaining the maximum and minimum value at first derivative level on the image. Equation 3.1 calculates the gradient operator, $\Delta$, and its application on vector $I$.

$$\Delta = \left( \frac{\partial}{\partial r}, \frac{\partial}{\partial c} \right) \quad \Delta I = \left( \frac{\partial I}{\partial r}, \frac{\partial I}{\partial c} \right)$$ (3.1)

The $\Delta I$ can then be used to find the value of the gradient magnitude $|\Delta I|$ and orientation $\phi$ of the image. The gradient magnitude shows the strength of an image edge and gradient orientation shows the edge pixel orientation. The classical gradient operators selected in this work are Sobel, Prewitt, Robert and Canny (Heath et al., 1997; Ziou and Tabbone, 1998).

Laplacian Operator is a second order derivative, where the value of edge pixel at the first derivative is referred to as zero-crossing at second order derivative. The disadvantage of this operator is its sensitive feature towards noise effect. In solving this problem, Gaussian function is being applied on the image. This is termed as Laplacian of Gaussian (LoG). The operators are explained below.
3.3.1. Sobel Operator

Sobel Operator uses a 3x3 convolution mask which is the x and y direction on the image. It is discovered at first derivative level. The horizontal and vertical pixel masks for Sobel Operator are shown in Figure 3.3.

\[
\begin{array}{ccc}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{array}
\quad
\begin{array}{ccc}
1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1 \\
\end{array}
\]

\( g^1 \quad g^2 \)

(a) (b)

Figure 3.3 : Sobel Operator

The mask will be moved until all the images and each value, \( R \), will be kept into an output array, which is located at the mask centre. The formula to find the gradient magnitude is Equation 3.2.

\[ |G| = |G_x| |G_y| \quad (3.2) \]

where \( G_x \) and \( G_y \) are given by the formulae 3.3 and 3.4 respectively.

\[ G_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6) \quad (3.3) \]
\[ G_y = (a_0 + ca_1 + a_2) - (a_6 + ca_5 + a_4) \quad (3.4) \]

where \( c \) is a constant with a value 2. Figure 3.4 shows the neighbourhood pixels that describe sobel operator concept.

\[
\begin{array}{ccc}
a_0 & a_1 & a_2 \\
a_7 & i, j & a_3 \\
a_6 & a_5 & a_4 \\
\end{array}
\]

Figure 3.4 : Neighbourhood pixel describe Sobel Operator Concept

3.3.2. Prewitt Operator

Prewitt Operator is based on the central difference concept and is given by
\[
\frac{\partial I}{\partial x} \approx \frac{I(x + 1, y) - I(x - 1, y)}{2}
\]  
(3.5)

This will produce a convolution mask (Figure 3.5),

\[
\begin{array}{ccc}
-1 & 0 & 1 \\
\end{array}
\]

**Figure 3.5 : Convolution Mask**

Prewitt Operator is much more sensitive to noise effect. Thus, an averaging process will be used to solve the noise problem. The convolution mask for Prewitt Operator has been implemented after averaging the process at x and y axis for \(\delta/\delta x\) and \(\delta/\delta y\). The equations for Prewitt Operator and Sobel Operator are quite similar except for the value of the constant \(c=1\).

### 3.3.3. Robert Operator

Robert Cross Operator uses 2x2 convolution masks. It uses \{+1,-1\} operator that will calculate the value

\[
I(x_i) - I(x_j)
\]  
(3.6)

for \((i,j)\) pixel at environs pixel. Mathematically, this equation is known as “forward differences”.

\[
\frac{\partial I}{\partial x} \approx I(x + 1, y) - I(x, y)
\]  
(3.7)

Convolution mask of Robert Cross Operator is illustrated in Figure 3.6.

\[
\begin{array}{ccc}
+1 & & \\
-1 & & +1 \\
\end{array}
g^1 
\begin{array}{ccc}
+1 & & \\
-1 & & +1 \\
\end{array}
g^2
\]

**Figure 3.6 : Convolution mask using Robert Cross Operator**
Calculation for gradient magnitude is given as:

\[ G = \sqrt{(g_1 * f)^2 + (g_2 * f)^2} \]  

(3.8)

### 3.3.4. LoG Operator

The Laplacian \( L(x,y) \) of an image with pixel intensity values \( I(x,y) \) is given by:

\[ L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \]  

(3.9)

Since the input image is represented as a set of discrete pixels, the discrete convolution kernel can approximate the second derivatives in the definition of the Laplacian. Three commonly used small kernels are shown in Figure 3.7.

<table>
<thead>
<tr>
<th>0 1 0</th>
<th>1 1 1</th>
<th>-1 2 -1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 -4 1</td>
<td>1 -8 1</td>
<td>2 -4 2</td>
</tr>
<tr>
<td>0 1 0</td>
<td>1 1 1</td>
<td>-1 2 -1</td>
</tr>
</tbody>
</table>

**Figure 3.7 : Discrete approximations to the Laplacian filter**

The application of Gaussian smoothening as a pre-processing step reduces the high frequency noise components prior to the differentiation step. The 2-D LoG function centered on zero and with Gaussian standard deviation \( \sigma \) has the form:

\[ \text{LoG}(x, y) = -\frac{1}{\pi\sigma^4} \left[ 1 - \frac{x^2 + y^2}{2\sigma^2} \right] e^{-\frac{x^2 + y^2}{2\sigma^2}} \]  

(3.10)

### 3.3.5. Canny Operator

The Canny technique is implemented as 5 separate steps, as given below.

1. Smoothing: Blurring of the image to remove noise.
2. Finding gradients: The edges should be marked where the gradients of the image has large magnitudes.
3. Non-maximum suppression: Only local maxima should be marked as edges.
4. Double thresholding: Potential edges are determined by thresholding.
5. Edge tracking by hysteresis: Final edges are determined by suppressing all edges that are not connected to a very certain (strong) edge.

The canny edge detector first smoothes the image, to eliminate noise and then finds the image gradient to highlight regions with high spatial derivatives. The algorithm then tracks along these regions and suppresses any pixel that is not at the maximum (non-maximum suppression). The gradient array is then further reduced by hysteresis. Hysteresis is used to track along the remaining pixels that have not been suppressed. Hysteresis uses two thresholds and if the magnitude is below the first threshold, it is set to zero (made a non-edge), else it is made an edge. If the magnitude is between the 2 thresholds (T1 and T2), then it is set to zero, unless there is a path from this pixel to a pixel with a gradient above T2.

3.3.6. Analysis of the Edge Detection Algorithms

The analysis was performed by visual inspection of the edge detection results and speed of the algorithms. The result here is presented for a sample partial fingerprint image. Similar results were obtained for other images also. The visual results, along with the speed of the algorithms, are presented in Figure 3.8.

From the results, it can be seen that the Canny operator produces the best edge detection result with respect to quality. As edge detection is the most important step of the proposed algorithm, the quality of detection was given preference and the canny operator was used to detect the edge regions during the preprocessing and segmentation steps.
3.4. PHASE 1: PREPROCESSING

The need of pre-processing arises because of the distortion and defect pixels of fingerprint imaging. It is a vital process of recognition system and is defined as an art that uses image processing algorithms to enhance the quality of an image. The goal of fingerprint image preprocessing is to

- Increase quality of the image
- Increase the interpretability of the image data
- Improve the segmentation and feature extraction processes.

The proposed enhanced VMF is shown in Figure 3.9.
The imbalances in brightness, contrast and color levels are automatically adjusted (Auto Imbalance Adjuster) and an enhanced vector median filter (VMF denoising) is proposed to denoise the input image. Brightness adjustment procedure automatically adjusts brightness imbalances of images to optimum level. Automatic contrast adjustment adjusts contrast imbalances of images to optimum level using histogram equalization. The color level adjustment procedure automatically adjusts the color levels of images to optimum levels. The VMF denoising procedure removes unwanted pixels that obscure important parts. The details regarding the Auto Imbalance Adjuster and VMF denoising procedures are described in the following sections.
3.4.1. Auto Imbalance Adjuster

The algorithm used to adjust the brightness, contrast and color level is presented in Figure 3.10. Three widely used color spaces are RGB (Red-Green-Blue), HSV (Hue-Saturation-Value) and NTSC (National Television Standards Committee). The first 3 steps of the proposed preprocessing algorithm work in NTSC color space. NTSC is the only color space among the above-mentioned ones that realizes a complete separation between the luminance and the chrominance information. NTSC has this property because, when it was introduced, it had to separate the information used by the monochrome TV receivers from the supplementary one used by color receivers (Blinn 1993). The components of the NTSC color space are Y (the luminance component), I (the cyan-orange component), and Q (the green-purple component). NTSC color space is used because it can portray the separation between the luminance and chrominance information in an effective manner and hence can separate intensity and color information into separate components. This is fully exploited in the proposed algorithm that automatically adjusts the contrast, brightness and color levels of an image.

The algorithm first converts the RGB color space into NTSC and performs various arithmetic operations and histogram equalization to correct the imbalances. After enhancement, the results are converted back to RGB color space. The output of the image imbalance adjuster is taken as input by the denoising algorithm.
3.4.2. VMF Denoising

Image noise is defined as the random variation of brightness or color information in images produced by medical devices or scanners. Image noise is generally regarded as an undesirable by-product during image acquisition. All images contain some visual noise. The presence of noise gives an image a mottled, grainy, textured or snowy appearance. These noises have to be...
removed or reduced for successful image processing applications like pattern recognition.

During image acquisition, inhomogeneities occur due to variance in relative position of the light source, camera position and the fingerprint position. These inhomogeneities make some part of the image appear darker and many have imbalanced contrast. Further, the images are often distorted by the presence of impulse noise, that appears as bright dots or dust particles and are normally distributed randomly all over the image.

Impulsive noise can be classified as Salt&Pepper Noise (SPN) and Random-Valued Impulse Noise (RVIN). An image containing impulsive noise can be described as follows:

\[
x(i,j) = \begin{cases} 
\eta(i,j) & \text{with probability } p \\
y(i,j) & \text{with probability } 1 - p
\end{cases}
\]

where \(x(i,j)\) denotes a noisy image pixel, \(y(i,j)\) denotes a noise free image pixel and \(\eta(i,j)\) denotes a noisy impulse at the location \((i,j)\). In salt-and-pepper noise, noisy pixels take either minimal or maximal values i.e. \(\eta(i,j) \in \{L_{\min}, L_{\max}\}\) and for random-valued impulse noise, noisy pixels take any value within the range minimal to maximal value, that is, \(\eta(i,j) \in [L_{\min}, L_{\max}]\) where \(L_{\min}\) and \(L_{\max}\) denote the lowest and the highest pixel luminance values within the dynamic range respectively. Thus, removing random valued impulse noise is more challenging than salt and pepper noise (Mallat and Hwang, 1992). The main difficulty that has to be faced for attenuation of noise is the preservation of image details. The difference between SPN and RVIN may be best described by Figure 3.11.

In the case of SPN, the pixel substitute in the form of noise may be either \(L_{\min}(0)\) or \(L_{\max}(255)\), whereas in RVIN situation, it may range from \(L_{\min}\) to \(L_{\max}\). Cleaning such noise is far more difficult than cleaning fixed-valued
impulse noise since for the latter, the differences in gray levels between a noisy pixel and its noise-free neighbors are significant most of the times.

![Salt and Pepper Noise](image1)

(a) Salt and Pepper Noise with $R_{ij} \in \{n_{\text{min}}, n_{\text{max}}\}$

![Random Valued Impulse Noise](image2)

(b) Random Valued Impulse Noise with $R_{ij} \in [n_{\text{min}}, n_{\text{max}}]$  

Figure 3.11 : Representation of Impulse Noise

The preprocessing phase presents a unified model termed as ‘Enhanced Vector Median Filter (EVMF)’ to solve the above preprocessing requirements and considers both random and salt-and-pepper noise.

3.4.3. Traditional Vector Median Filter

All denoising methods depend on a filtering parameter ‘$h$’. This parameter measures the degree of filtering applied to the image. For most methods, the parameter ‘$h$’ depends on an estimation of the noise variance $\sigma^2$.

The result of a denoising method $D_h$ can be defined as a decomposition of any image ‘$v$’ as given in Equation (3.12).

$$w = D_h v + n(D_h, v)$$  \hspace{1cm} (3.12)

where

1. $D_h v$ is more smooth than $v$
2. $n(D_h, v)$ is the noise guessed by the method.

In the present environment, it is needed to make sure that the denoising method smoothenes $v$ and also to make sure that the contents lost due to noise are recovered (Malgouyres, 2001; Osher et al., 2004). For this purpose, several
types of noise reduction techniques are available. They include, median filter (Pitas and Venetsanopoulos, 1990; Church et al., 2008), Vector Median Filter (VMF) (Astola et al., 1990; Laskar et al., 2009), Anisotropic diffusion Filter (Fu et al., 2006) and Wavelet Based Filters (Ibrahim et al., 2007).

Median filter, a non-linear filtering technique, uses a window that moves over a signal and at each point, the median value of the data within the window is taken as the output. The impulse response of the median filter is zero and thus makes its use attractive to suppress impulsive noise. Median filters are robust and are well-suited for data smoothing when the noise characteristics are not known and also has the capability to preserve edges. Median filtering works very well for impulse noise removal in grey scale images, but with three channels (Red, Green and Blue), the correlation amongst them also needs to be considered.

Using these vector signal properties, the VMF are built. In the vector median approach, the samples of the vector-valued input signal are processed as vectors as opposed to component-wise scalar processing. The vector median operation inherently utilizes the correlation between the signal components giving the filters some desirable properties.

A vector median is defined as the vector that corresponds to the minimum sum of distances to all other vector pixels. The selection of the pixel with minimum sum of distance may be readily visualized as finding the pixel nearest to the ‘centre’ of the pixels within the neighborhood viewed as a cluster in the RGB space.

The algorithm has three main steps. The first step, after dividing an image into fixed-equal sized windows, computes the Euclidean distance from every pixel to every other pixel in its neighborhood in the current window chosen. The algorithm, in the next step, arranges the vector pixels of this window in ascending order on the basis of the sum of distances. The ordering used for sum of distances is associated with the vector pixels also. The vector
pixel with the smallest sum of distances is the vector median pixel. The vector median filter is represented as

$$X_{VMF} = \text{vectormedian(window)}$$  \hspace{1cm} (3.13)

If $$\delta_i$$ is the sum of the distances of the $$i$$th vector pixel with all the other vectors in the kernel, then

$$\delta_i = \sum_{n=1}^{N} d(x_i, x_n)$$  \hspace{1cm} (3.14)

where $$d(X_i, X_n)$$ represents a distance measure between the $$i$$th and the $$n$$th neighboring vector pixels with ($$1 \leq i < N$$) and $$X_i$$ and $$X_N$$ are vectors with $$N=9$$. The ordering may be illustrated using Equation (3.15) and this implies the same ordering to the corresponding vector pixels (Equation 3.16). In these equations, the subscripts represent the ranks.

$$\delta_{(1)} \leq \delta_{(2)} \leq \ldots \leq \delta_{(9)}$$  \hspace{1cm} (3.15)

$$x_{(1)} \leq x_{(2)} \leq \ldots \leq x_{(9)}$$  \hspace{1cm} (3.16)

Since the vector pixel with the smallest sum of distances is the vector median pixel, it will correspond to rank 1 of the ordered pixels (Equation 3.17).

$$X_{VMF} = X(1)$$  \hspace{1cm} (3.17)

The steps are consolidated in Figure 3.13.

The VMF is highly effective in removing impulsive noise but also has the following disadvantages.

- It fails to distinguish thin lines and boundaries from impulsive noise and usually filters them out because it interprets these fine details as some noise.
High computation cost – due to repeated distance calculation of similar values in the filtering window. These can be removed to reduce calculation cost.

More than one pixel derives the minimum distance thereby more than one qualified pixel to replace the center pixel.

The non-unequivocal mathematical distance calculation ignores the difference between the three color spaces leading to color distortions.

In this phase of the study, solutions to solve these issues are presented.

### 3.4.4. Enhanced VMF

The enhanced VMF uses a classification procedure to differentiate edges and boundaries and apply VMF only to other regions, thus the algorithm solves the problem of filtering edge details. The algorithm uses a fast VMF variant to reduce the computation cost. The problem of more than one pixel having the same distance and color distortion introduced are solved by using a perception referred criteria and using a different color space model. The color space model used is the HSI (Hue, Saturation and Intensity) for the selection of the ‘most suitable’ candidate. The enhanced algorithm consists of the following steps.

1. Apply Region Analyzer
2. Impulse noise detection on non-edge region blocks
3. Denoising using fast VMF on detected noise
a) Region analyzer

The algorithm divides the image into 16 x 16 blocks and then applies region analyzer to classify the image into edges and non-edge regions. The region analyzer uses the Canny edge detector for this purpose. The procedure begins by dividing the input image into 4 x 4 blocks. The Canny edge operator is then used on each block to detect the edges. When an original pixel O(i, j) at a block is computed, the values of the original pixels nearby O(i, j) are used to calculate the gradients in the horizontal and vertical directions using Equation 3.18.

\[
G_x = [O(i-1, j-1) + 2O(i-1, j) + O(i-1, j+1)] - [O(i+1, j-1) + 2O(i+1,j) + O(i+1, j+1)]
\]

\[
G_y = [O(i-1, j-1) + 2O(i, j-1) + O(i+1, j-1)] - [O(i-1, j+1) + 2O(i,j+1) + O(i+1, j+1)]
\]

(3.18)

If the \( G_x \) or \( G_y \) is larger than the threshold, then the pixel is classified as a portion of an edge, else it is treated as a non-edge pixel. The threshold value is set as 15 during experimentation. The edge portion is termed as ‘Edge Region’ and the non-edge pixels are termed as ‘Detailed Region’.

b) Impulse Noise Detection

The denoising part of the algorithm is a two-step process, detection step and filtering step and is applied only to the non-edge region. The noise detection step uses gradient values (Ville et al., 2001; Nachtegaele et al., 2002) to determine whether a pixel is corrupted with noise or not. For each pixel, a 3 x 3 window surrounding the pixel is constructed (Figure 3.13) and a novel rule-based method is used during noise detection. Each neighbor is related to its corresponding direction \{NW = north west, N = north, NE = north east, W = west, E = east, SW = south west, S = south, SE = south east\} and position (indicated in suffix). If the non-edge region is represented as A, then the gradient \( G_{k,j}(A_{i,j}) \) is calculated using Equation 3.19.

\[
G_{k,j}(A_{i,j}) = A(i + k, j + l) - A(i, j)
\]

(3.19)
where \( k, l \in \{-1, 0, 1\} \) and the pair of \((k,l)\) corresponds to one of the eight directions and is called the center of the gradient. The eight gradient values, corresponding to the eight different directions or neighbors, are called the basic gradient values and are used to determine if a central pixel is corrupted with impulse noise or not. This is determined by analyzing the gradient value. A large gradient value can arise in three conditions.

1. Central pixel is not noisy, but its neighboring pixels are noisy
2. The central pixel is affected by impulse noise.
3. A natural large gradient has occurred in the image because of strong edges.

![Figure 3.13: Central Pixel and its Neighbouring Pixels](image)

Identification of noisy neighbourhood is determined using the eight gradient values obtained from eight directions. The first situation is determined by considering each neighbouring pixel (basic) and comparing it with two related gradients for each direction. The two related gradient values in the same directions are determined using the center which are in right angle direction to the basic. Consider Figure 3.14 as an example. Here, for NW direction, the basic gradient value is calculated along with the two related gradient values. When all these gradient values are small, then \((i, j)\) is considered as noisy.
Table 3.1 shows the basic and its corresponding related pixels for each direction.

**TABLE 3.1**

**BASIC AND RELATED PIXELS**

<table>
<thead>
<tr>
<th>Direction</th>
<th>Basic</th>
<th>Related</th>
</tr>
</thead>
<tbody>
<tr>
<td>NW</td>
<td>$G_{NW}A(i, j)$</td>
<td>$G_{NW}A(i+1, j-1), G_{NW}A(i-1, j+1)$</td>
</tr>
<tr>
<td>N</td>
<td>$G_{N}A(i, j)$</td>
<td>$G_{N}A(i, j-1), G_{N}A(i, j+1)$</td>
</tr>
<tr>
<td>NE</td>
<td>$G_{NE}A(i, j)$</td>
<td>$G_{NE}A(i-1, j-1), G_{NE}A(i+1, j+1)$</td>
</tr>
<tr>
<td>E</td>
<td>$G_{E}A(i, j)$</td>
<td>$G_{E}A(i-1, j), G_{E}A(i+1, j)$</td>
</tr>
<tr>
<td>SE</td>
<td>$G_{SE}A(i, j)$</td>
<td>$G_{SE}A(i-1, j+1), G_{SE}A(i+1, j-1)$</td>
</tr>
<tr>
<td>S</td>
<td>$G_{S}A(i, j)$</td>
<td>$G_{S}A(i, j-1), G_{S}A(i, j+1)$</td>
</tr>
<tr>
<td>SW</td>
<td>$G_{SW}A(i, j)$</td>
<td>$G_{SW}A(i-1, j-1), G_{SW}A(i+1, j+1)$</td>
</tr>
<tr>
<td>W</td>
<td>$G_{W}A(i, j)$</td>
<td>$G_{W}A(i-1, j), G_{W}A(i+1, j)$</td>
</tr>
</tbody>
</table>

While considering the second condition, if the gradient of more than 4 neighboring pixels (more than half the number of pixels) is large, then the central pixel is considered as impulse noise. If the first two conditions fail, then all the pixels in that block are considered to be noise free. The large and small threshold values are obtained.
c) Filtering Process

Once identifying the impulse noise regions in an image, the next step is to remove them using Vector Median Filter. As mentioned earlier, one major drawback of VMF is its high repeated computation cost. This is solved by performing VMF only on the noisy pixels identified. This makes VMF faster for high noise density images. The speed difference is more obvious with high noise density images than with images having low impulse noise. In order to solve the ambiguity of more than one pixel having the same minimum distance and color distortions introduced by the traditional VMF, the following modifications are made to the final step of the VMF algorithm.

\[
V = \text{Minimum Distance (R Color component of neighbour Pixels)}
\]

If \( V > 1 \) then

\[
V = \text{Minimum Distance (R, G Color components of Pixels)}
\]

If \( V < 1 \) then

\[
V = \text{Minimum Distance (R, G, B Color components of Pixel)}
\]

\[
\text{Vector Median} = V
\]

Inclusion of the above procedure avoids the problem of ambiguity and further speeds up the algorithm. The functioning of the algorithm is consolidated in Figure 3.15.

The proposed enhancement preprocessor was tested with several test images. The quantitative and statistical results are presented in Chapter 4 (Results and Discussion), Section 4.2.
3.5. PHASE II: SEGMENTATION

An important step in automatic fingerprint recognition is the segmentation of fingerprint. Segmentation is defined as the decomposition of an image into its components. In the proposed partial fingerprint recognition system, segmentation refers to ROI-segmentation (Region Of Interest) where the aim is to partition an input image into two regions, foreground and background.
background. The foreground represents the fingerprint and the background represents the rest of the image. The task here is to decide which part of the image belongs to foreground, originating from the contacts of a fingerprint with the sensor and which part belongs to background. The main focus of ROI-segmentation is to accurately extract the ROI, that is, the fingerprint, so that the process of feature extraction and matching can be performed accurately. Correct segmentation removes background which consequently removes unwanted pixels, thus reducing the number of false features extracted. Examples of segmentation results are shown in Figure 3.16.

![Figure 3.16: Examples of ROI](image)

The proposed segmentation method was developed to achieve three goals:

i. improve the accuracy of feature detection

ii. reduce the time of subsequent processing so as to enhance the performance of the entire system.

iii. remove the background region and retain the area of the fingerprint image.

For this purpose, image segmentation based on edge and corner detection is proposed and analyzed. Corners are local image features characterized by locations where variations of intensity function $f(x, y)$ in both X and Y directions are high. It is defined as the intersection points between two or more edge segments. It combines sudden intensity changes and sudden change in the direction of edge track during corner and edge detection process.
for segmentation. The basic idea used by the corner detection algorithms is to find points where two edges meet (Figure 3.17).

![Figure 3.17 : Example of Corners in an Image](image)

Two most frequently used corner detectors are Harris Corner and Susan Corner algorithms. Harris method works by comparing the corner strength based on a local structure matrix and SUSAN (Smallest Univalue Segment Assimilating Nucleus) detector is based on brightness comparison. Both algorithms work with the same basis, that is, when a calculated value of the mentioned detectors (which is characteristic for a corner) exceeds a given threshold, the processed image point is detected as a corner. Irrespective of the detection method used, the algorithm should satisfy the following criteria.

- Only the true corners should be detected.
- Corner points should be well localized.
- Corner detector should be robust with respect to noise.
- Corner detector should be efficient in terms of time.

Both the algorithms detect edges and corners by following a simple procedure. The procedure starts by constructing a small window (3x3, 5x5, 7x7) and moving the window while analyzing the intensity changes in all directions. If there is no change, then it is considered as a flat region; when the change occurs only in one direction, then it is considered as an edge region. If the change is envisaged in all directions, then it is considered as a corner. After detecting the corners, the strength of the detected edges and corners are
calculated. All edges and corners whose strength is greater than a threshold are reported as the actual detected corners. Detailed description of both these algorithms is given in the following sections.

3.5.1. Susan-Corner Detection

SUSAN corner detector is realized by a circular mask (Smith and Brady, 1997). Consider Figure 3.18, showing a dark rectangle on a white background, five circular masks (having a centre pixel known as nucleus) at different positions on the simple image. Corners can be detected according to the area of USAN (Univalue Segment Assimilating Nucleus). If the brightness of each pixel within a mask is compared with the brightness of that mask's nucleus, then an area of the mask can be defined which has the same (or similar) brightness as the nucleus. This area of the mask is known as the “USAN”). In Figure 3.19, each mask from Figure 3.18 is depicted with its USAN shown in white.

In order to detect corners, the similar comparison function between each pixel within the mask and mask's nucleus is given by Equation 3.20.

\[
c(r, r_o) = \exp\left\{ -\left( \frac{I(r) - I(r_o)}{t} \right)^6 \right\}
\]

(3.20)

where \(r_o\) is the nucleus’s coordinate and ‘r’ is the coordinate of other points within the mask, \(c(r, r_o)\) is the comparison result, \(I(r)\) is the point’s gray value, \(t\) is gray difference threshold which determines the anti-noise ability and the smallest contrast that can be detected by SUSAN detector. The size of USAN region is calculated using Equation (3.21).

\[
n(r_o) = \sum_{r \in c(r_o)} c(r, r_o)
\]

(3.21)
The initial response to corners is obtained using Equation (3.22), which is in accord with the principle of SUSAN, that is, the smaller the USAN region, the greater to the initial response to corners.

$$R(r_0) = \begin{cases} g - n(r_0) & n(r) < g \\ 0 & n(r) \geq g \end{cases} \quad (3.22)$$
In the above equation, \( g \) is the geometric threshold, which determines the acute level of a corner, the smaller the acuter, the more accurate is the corner information. Finally, the corners can be found by non-maximum inhibition. The Susan algorithm is summarized in Figure 3.20.

| **Input Image and determine a circular mask (37 pixels radius) around a nucleus for each point within the image** |
| **Calculate the difference in brightness between each pixel of the mask and that of its nucleus and** |
| \[
  c(r, r_0) = \begin{cases} 
  1 & \text{if } |I(r) - I(r_0)| < g \\
  0 & \text{Otherwise}
  \end{cases}
\] |
| where \( r \) is a given pixel location within the mask and \( r_0 \) is the position of the mask nucleus in the image frame. \( I(r) \) refers to the intensity of the pixel at location \( r \), \( t \) is the brightness threshold and \( c \) is the output of the comparison in intensities at locations \( r \) and \( r_0 \). |
| **Sum the number of pixels within the circular mask which have similar intensity levels to that of the nucleus** |
| \[
  n(r_0) = \sum c(r, r_0)
\] |
| The sum of the comparison outputs (\( c \)) is then taken and that represents the total number of pixels in the USAN region. In other words, USAN area is the corner response. |
| **Calculate geometric threshold as half of the maximum value of \( n \) (\( n_{\text{max}}/2 \))** |
| **Compare \( n \) with \( g \) to detect perfect corner. At a perfect corner, the USAN area will always be less than half the size of the mask area and will be a local minimum** |
| \[
  R(r_0) = \begin{cases} 
  g - n(r_0) & \text{if } n(r_0) < g \\
  0 & \text{Otherwise}
  \end{cases}
\] |

**Figure 3.20 : Susan Algorithm**
3.5.2. Harris corner detection algorithm

Harris corner detection algorithm is realized by calculating each pixel’s gradient (Harris and Stephens, 1988). If the absolute gradient values in two directions are both high, then the pixel is judged as a corner. Harris corner detector is defined as follows:

\[ R = \text{det}(M) - k \text{tr}^2(M) \]  \hspace{1cm} (3.23)

where

\[ M(x, y) = \begin{bmatrix} I_u^2(x, y) & I_{uv}(x, y) \\ I_{uv}(x, y) & I_v^2(x, y) \end{bmatrix} \]

\[ I_u^2(x, y) = X^2 \otimes h(x, y), \quad I_v^2(x, y) = Y^2 \otimes h(x, y), \quad I_{uv}(x, y) = XY \otimes h(x, y) \]

\[ h(x, y) = \frac{1}{2\pi} e^{-\frac{x^2+y^2}{2}} \]

where \( I_u(x, y) \) and \( I_v(x, y) \) are the partial derivatives of the gray scale in direction \( u \) and \( v \) at point \((x, y)\) and \( I_{uv}(x, y) \) is the second-order mixed partial derivative, \( k \) is an empirical value, \( h(x, y) \) is a Gaussian function, \( X \) and \( Y \) are the first-order directional differentials, which can be approximately calculated by convolving the gray scale and difference operators in direction \( u \) and \( v \). Gaussian function is used to reduce the impact of noise, because first-order directional differentials are sensitive to noise. If \( R \) exceeds a certain threshold, then the point is taken as a corner.

The steps used by Harris and Susan corner detection is summarized in Figure 3.21.
Figure 3.21: General Steps in Corner Detection

3.5.3. Segmentation using Corner Detection Algorithms

The segmentation algorithm using corner detection algorithm is given in Figure 3.22.

- Divide image into blocks
- Identify set of corners for each block (Block size - 9 x 9)
- Create Binary image using adaptive threshold operation and select only those corners above threshold.
  - Threshold = 0.025 x (max. Corner Strength)
  - Perform dilation to fill in holes or gaps (D)
- Detect Region of Interest (ROI) by using mask 1 and mask 2 where Threshold = c x size of window^2 (0 > c < 1).
  - Mask 1: Set C to a very high value to reduce the size of noisy segments and ignore boundary areas (C=0.9 used during experimentation)
  - Mask 2: Set C to a low value to remove noisy segments (C=0.4 used during experimentation)

Figure 3.22: Corner Detection based Segmentation

During experimentation, it was found that the above algorithm produces false corners. To solve this problem, the above procedure is combined with Canny edge detector. Canny edge detecting algorithm was chosen because of its efficient performance (Section 3.3). The modified proposed image
segmentation based on edge and corner detection consists of the following steps.

Step 1: Read input Image  
Step 2: Create mask window (block for Harris and circle for Susan)  
Step 3: Perform Canny Edge Detection on window  
Step 4: Perform Harris / Susan segmentation over the detected edges alone.

The proposed algorithms, Susan-based and Harris-based, have the advantages of being accurate, resistant to noise and faster than the traditional methods. The experimental results are presented in the next chapter, Results and Discussion.

3.6. PHASE III: FEATURE EXTRACTION

Extraction of important and useful features of interest from partial and full fingerprint images is a crucial task during recognition. Feature extraction algorithms select only relevant features important for increasing the recognition rate and results with “feature vector”. The feature extraction algorithm should take the following points into account.

- The features should carry enough information about the fingerprint and should not require any domain-specific knowledge for their extraction.
- They should be easy to compute in order to attain a feasible approach for a large image collection.

Many feature extraction techniques have been developed, which focus to bring the above points. Both level 1 and level 2 features are examples of such type. While considering partial fingerprint matching, however, the feature selection and extraction have to be given more importance, as the selected features should help to maximize small print-large print matching performance. In the present research work, for this purpose, LBP around pore feature and SIFT are used.
Using local image features for recognition is a scheme that has gained attention in the recent years. Here, the feature detection algorithm first detects regions that are covariant to a class of transformations and then for each detected region, an invariant descriptor is built. These descriptors are treated as features and are used during matching. It has been used in many computer vision applications, like image retrieval, wide baseline matching, object recognition, texture recognition, and robot localization (Mikolajczyk and Schmid, 2005). While using local features, the region detection is the critical step. Several techniques have been proposed, all of which emphasize different image properties such as pixel intensities, color, texture, and edges. Examples include corner detection algorithms for region detection, blobs or distribution based techniques like histograms. In the present research work, two such features, namely, Local Binary Pattern (LBP) and SIFT are used.

The SIFT descriptor (Lowe, 2004) is a 3D histogram of gradient locations and orientations where the contribution to the location and orientation bins is weighted by the gradient magnitude and a Gaussian window overlaid over the region. The LBP is a nonparametric and computationally simple descriptor of local texture patterns. Here, the histogram of the binary patterns computed over a region is used for texture description. Both the selected descriptors have shown successful results with several state-of-the-art applications like face recognition, background subtraction and recognition of 3D textured surfaces (Heikkil et al., 2009).

SIFT and LBP exhibit high tolerance to illumination changes, perspective distortions, image blur and image zoom, and are also very robust to occlusion. Pore features have the advantage that in spite of the small number can uniquely and accurately identify fingerprints. SIFT features generate a large number of features over a broad range of scales and locations. The number of SIFT feature points can be regulated by a set of parameters such as the number of octaves and scales. Performance of SIFT features on the number
of feature points is very similar to that of minutiae. Because of these advantages, it was decided to use them during feature extraction and matching.

3.6.1. Pore Extraction

Pores in a fingerprint can be of two types. They are open and closed pores (Refer to Chapter 1, Figure 1.11 for example). Two algorithms are proposed to extract both types of pores in a fingerprint. The closed pores are extracted using a marker controlled watershed segmentation algorithm and the open pores are extracted using a sequence of thinning and spurring operations.

- Closed Pore Extraction
  a. Watershed Transform

The watersheds concept is one of the classic tools in the field of topography and was introduced by Beucher and Lantuejoul (1979). It consists of placing a water source in each regional minimum, to flood the relief from sources, and build barriers when different sources are meeting. The resulting set of barriers constitutes a watershed by flooding. It is the line that determines where a drop of water will fall into a particular region.

Mathematically, in the given topographic representation of a given image $I$, the intensity value of each pixel stands for the elevation at this point. If the image $f$ is an element of the space $C(D)$ of a connected domain $D$ then the topographical distance between points $p$ and $q$ in $D$ is,

$$ T_f(p,q) = \inf_{\gamma} \int \Delta f(\gamma(s)) \, ds $$

(3.24)

where $\inf_{\gamma}$ is over all paths (smooth curve) inside $D$. Using this, Roerdink and Meijster (2001) define the watershed as given below.

Let $f \in C(D)$ have a minima $\{m_k\}_{k \in I}$, for some index set $I$. CB $(m_i)$ of a minimum $m_i$ is defined as the set of points. $C \in D$, which are topographically closer to $m_i$ than to any other regional minimum $m_j$. 
\[ CB(m_i) = \{ x \in D \mid \forall j \in I \setminus \{ i \} : f(m_i) + T_f(x,m_i) < f(m_j) + T_f(x,m_j) \} \]  

(3.25)

The watershed of \( f \) is the set of points that do not belong to any catchment basin

\[ W_{\text{shed}}(f) = D \cap \left( \bigcup_{i \in I} CB(m_i) \right) \]  

(3.26)

Let \( W \) be some label and \( W \in I \). The watershed transform of \( f \) is a mapping of \( \lambda : D \to I \cup \{ W \} \) such that \( \lambda(p) = i \) if \( p \in CB(m_i) \) and \( \lambda(p) = W \) if \( p \in W_{\text{shed}}(f) \). So the watershed transform of \( f \) assigns labels to the points \( D \), such that

(i) different catchment basins are uniquely labeled and
(ii) a special label \( W \) is assigned to all points of the watershed of \( f \).

The advantage of the watershed transform is that it produces closed and adjacent contours including all image edges. However, often the watershed produces a severe over-segmentation also. Some solutions of the over-segmentation are addressed by Meyer and Beucher (1990).

In image processing, different watershed lines may be computed. Here a gray-scale image is considered as topographic surface. If this surface is flooded from its minima and if the merging of the waters coming from different sources is prevented, then the image can be partitioned into two different sets: the catchment basins and the watershed lines. When this transformation is applied to the image gradient, the catchment basins correspond to the homogeneous gray level regions of this image.

b. **Marker controlled watershed segmentation**

A major enhancement of the watershed transformation consists in flooding the topographic surface from a previously defined set of markers. This prevents the over-segmentation problem of watersheds. The marker-controlled watershed segmentation has been shown to be a robust and flexible method for segmentation of objects with closed contours, where the boundaries are
expressed as ridges (Nguyen et al., 2003). Markers are placed inside an object of interest; internal markers associate with objects of interest, and external markers associate with the background. After segmentation, the boundaries of the watershed regions are arranged on the desired ridges, thus separating each object from its neighbors.

Creating Markers

The marker image used for watershed segmentation is a binary image consisting of either single marker points or larger marker regions, where each connected marker is placed inside an object of interest. Each initial marker has a one-to-one relationship to a specific watershed region, thus the number of markers will be equal to the final number of watershed regions. After segmentation, the boundaries of the watershed regions are arranged on the desired ridges, thus separating each object from its neighbors. The markers can be manually or automatically selected, but high throughput experiments often employ automatically generated markers to save human time and resources.

Closed Pores

In this work, a simple algorithm is used to create the foreground and background markers using morphological image reconstructions. The watershed transform of the gradient fingerprint image is computed without any other processing. The result is severely over-segmented due to the large number of regional minima as shown Figure 3.24b. By computing the location of all regional minima in the fingerprint image Figure 3.24c, it can be seen that most of the regional minima are very shallow and represent detail that is irrelevant to the segmentation problem. These extraneous minima are eliminated by computing the set of low spots in the image that are deeper (using a height threshold = 2) than their immediate surroundings. Then the markers are superimposed on the original fingerprint image.
The next step during pore extraction creates the background markers. The main aim here is to mark the background by finding pixels that are exactly midway between the internal markers. This is done by computing the watershed transform of the internal marker image. The resulting watershed ridgelines appear in midway between the pores and hence they serve well as external markers. The marker image is shown in Figure 3.24d.

The internal and external markers are then used to modify the gradient fingerprint image using a procedure called minima imposition. The minima imposition technique modifies a fingerprint image so that regional minima occur only in marked locations. Other pixel values are pushed up as necessary to remove all other regional minima. The gradient fingerprint image is then modified by imposing regional minima at the locations of both the internal and the external markers. Finally the watershed transform of the marker-modified gradient fingerprint image is computed. Superimposing the watershed ridgelines on the original fingerprint image produces improved pore extraction (Figure 3.24e). The steps during pore extraction using marker control watershed segmentation are given in Figure 3.23. An example of pore extraction when the proposed algorithm is applied to a partial fingerprint (5%) enlarged to 400% is shown in Figure 3.25. From Figures 3.24 and 3.25, it could be seen that the proposed pore extraction algorithm is efficient in extracting pores with both full fingerprint and even a very small partial fingerprint.

- **Open Pore Extraction**

  The steps involved while extracting the open pores in a fingerprint image are given below.

  Step 1 : Binarize image using a local threshold method.
  Step 2 : Thin the valleys of the binary image to obtain the dual skeleton.
  Step 3 : Extract the spurs present in the dual skeleton and discard all spurs which satisfy the following scenarios.
- The spur termination belongs to the valleys area, or
- The numbers of bright pixels around the spur termination is below a threshold.

- Read the gray-scale image.
- Develop gradient fingerprint images using edge detection function.
- Compute the watershed transform of the gradient fingerprint image without any other processing
- Calculate the regional minima to obtain the good forward markers
- Superimpose the foreground marker image on binarized fingerprint image.
- Clean the edges of the markers using edge reconstruction.
- Compute the background markers
- Compute the watershed transform of the function

\[ \begin{align*}
T &= \arg \max_T R(T) = \arg \max_T \frac{P(T)(1 - P(T))(\mu_f - \mu_b)^2}{P(T)\sigma_f^2 + (1 - P(T))\sigma_b^2} \\
&= \text{maximized } R(T)
\end{align*} \]

where \( P(T) = \sum_{i=0}^{T} p(i) \) is the cumulative distribution function of the histogram \( p(T) \), \( \mu_f = \sum_{i=0}^{T} ip(i) \) and \( \mu_b = \sum_{i=T+1}^{255} ip(i) \) are the means and
\( \sigma_f^2 = \sum_{i=0}^{T}(i - \mu_f)^2 p(i) \) and \( \sigma_b^2 = \sum_{i=T+1}^{255}(i - \mu_b)^2 p(i) \) are the variance of the foreground and background respectively. The maximized \( R(T) \) indicates how

Figure 3.23: Pore Extraction Algorithm

In the first step, the image is binarized using Otsu’s binarization algorithm (Otsu, 1979). This algorithm is used to determine the optimal threshold value for the image. The advantage of Otsu’s algorithm is that it maximizes the ratio of between-class variance/within-class variance. The threshold value is computed using Equation (3.27).
well the two classes are separated. A larger $R(T)$ implies a higher SNR level, which translates to a smaller classification error. Once the threshold is determined, each pixel in the region is classified as a member of black or white.

<table>
<thead>
<tr>
<th>(a) Original Partial Fingerprint</th>
<th>(b) Over-Segmented Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c) Regional Minima</td>
<td>(d) Marker Image</td>
</tr>
<tr>
<td>(e) Extracted Pores</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3.24: Pore Extraction Process**

The second step of the algorithm uses a thinning procedure to obtain the skeleton, which erodes the ridges step by step until they are 1 pixel wide. The
morphological operator suggested by Kamei and Mizoguchi (1995) is used during this step. The results after thinning are shown in Figure 3.26. After skeletonization, the open pores are extracted using the same procedure as with closed pore extraction method described in the previous section. An example is shown in Figure 3.27.

Figure 3.25 : Pore Extraction in Partial Fingerprint

Figure 3.26 : Example of Thinned and Skeleton Partial Fingerprint
3.6.2. Local Binary Patterns (LBP)

The LBP operator is a powerful means for describing texture. It is defined as a gray-scale invariant texture measure, derived from a general definition of texture in a local neighbourhood. For a given size of neighbourhood (3x3), a binary pattern is extracted at each pixel. Neighbors of the center pixel are threshold by the value of the center pixel, from which a binary pattern is obtained. This pattern is converted to sequence of bits by starting from a defined angle. In a (P,R) neighborhood, a binary code that describes the local texture pattern is built by thresholding of the gray value from its center. The binary code, that is, the LBP operator is denoted as LBP(P,R). The notion (P,R) means that there are P sampling points on a circle of radius R. Figure 3.28 illustrates how the LBP operator is computed in the (8,1) square neighborhood.

Figure 3.28: LBP Operator Computing in a (8, 1) Square Neighborhood
The binary code of this example is \((01011011)_b\). Thus, the LBP operator at \(g_c\) can be defined as:

\[
LBP(p, R) = \sum_{p=0}^{p-1} s(g_p - g_c)^2
\]

where \(s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{Otherwise} \end{cases}\), \(g_c\) corresponds to the graylevel of the center pixel of a local neighborhood and \(g_p\) to the graylevels of \(g\) equally spaced pixels on a circle of radius \(R\). Since the correlation between pixels decreases with distance, a lot of the texture information can be obtained from local neighborhoods. Thus, the radius \(R\) is usually kept small. In practice, Equation 3.28 means that the signs of the differences in a neighborhood are interpreted as an N-bit binary number, resulting in \(2^N\) distinct values for the binary pattern. From the above, it is easy to figure out that the LBP has several properties that favor its usage in interest region description. The features have proven to be robust against illumination changes, they are very fast to compute, and do not require many parameters to be set.

### 3.6.3. LBP Extraction around Pore Feature

Extracted pores are used as anchor points for mapping to full image. For each pore, a sub window (32x32) is formed centered at that pore. Each window is convolved with 8 gabor filters corresponding to scale = 1,2 and orientation = 0,30,60,135 and 8 gabor images are obtained at each pore. Each gabor image is convolved with rotation invariant LBP operator. Then 8 LBP histograms are captured for each pore location. These 8 LBP histograms are concatenated and resultant histogram is stored in the template image. Average LBP histogram of pores is shown in Figure 3.29. Thus finally in the template for the partial image will have a set of histograms corresponding to all pores.
3.6.4. SIFT Feature

In this study, for the purpose of extending characteristic feature points of fingerprint beyond minutiae points, a Scale Invariant Feature Transformation (SIFT) feature selection method is adopted. SIFT extracts repeatable characteristic feature points from an image and generates descriptors representing the texture around the feature points. In this work, the utility of SIFT representation for fingerprint-based identification for partial prints is studied. Since the SIFT feature points have already demonstrated their efficacy in other generic object recognition problems, it is also considered that this representation is stable and reliable for many of the matching problems related to the fingerprint domain. Further, since SIFT feature points are based on texture analysis of the entire scale space, it is hoped that these feature points will be robust to the fingerprint quality and deformation variation. The standard SIFT keypoint descriptor representation is advantageous in several respects.

(1) The representation being carefully designed to avoid problems due to boundary effects, smooth changes in location, orientation and scale do not cause radical changes in the feature vector.
(2) It is fairly compact, expressing the patch of pixels using a 128 element vector.
(3) While not explicitly invariant to affine transformations, the representation is surprisingly resilient to deformations such as those caused by perspective effects.
These characteristics are evidenced in excellent matching performance against competing algorithms (Ke and Sukthankar, 2004) and therefore SIFT is used in this study.

Minutiae points are strictly defined by the ridge ending and bifurcation points. Therefore, the number of minutiae points appearing in a fingerprint image is limited to a small number (<100). However, SIFT points are only limited by the condition of local minima or maxima in a given scale space, resulting in a large number of feature points. The number of SIFT feature points are affected by a set of parameters such as the number of octaves and scales. Typical fingerprints may contain up to a few thousand SIFT feature points. Figure 3.30 shows an example of minutiae points and SIFT feature points on the same fingerprint image. There are only 36 minutiae points, but the number of SIFT feature points is observed to be 2,020.

![Minutiae and SIFT feature points](image)

**Figure 3.30 : Minutiae and SIFT feature points**

The proposed method for SIFT extraction from fingerprint images is shown in Figure 3.31.
a) Preprocessing

Even though SIFT was originally developed for general purpose object recognition and does not require image pre-processing, to improve its efficiency a two-step preprocessing was performed.

i) Adjusting the gray level distribution

ii) Removing noisy SIFT feature points.

The first step is used to overcome the differences in grey level. Similarly, the images in the database and input image might be affected by different density of noise. It is a known fact that the performance of the SIFT extraction improves when the fingerprints have similar texture. Because of this, the preprocessing introduced in Section 3.3 was first performed to adjust contrast, brightness, gray-level and to remove noise.

The second point arises because the boundary area of a fingerprint always causes some feature points to be detected because they are local extrema. However, the boundary region is different for every fingerprint impression even for the same finger. Therefore, feature points on the fingerprint boundary usually result in false matches. Therefore, a binary mask
that includes only the inner part of a fingerprint is constructed and are used to prevent any noisy feature points from being detected on the boundary.

b) SIFT Feature Extraction Algorithm

Scale Invariant Feature Transformation (SIFT) was originally developed for general purpose object recognition. SIFT detects stable feature points in an image and performs matching based on the descriptor representing each feature point. A brief description of the SIFT operator is provided below.

Step 1: Scale Space Construction

A scale space is constructed by applying a variable scale Gaussian operator on an input image. Difference of Gaussian (DOG) images are obtained by subtracting subsequent scales in each octave. The set of Gaussian-smoothed images and DOG images are called an octave. A set of such octaves is constructed by successively down sampling the original image.

Step 2: Local Extrema

Local extrema are detected by observing each image point in DOG space. A point is decided as a local minimum or maximum when its value is smaller or larger than all its surrounding neighboring points by a certain amount.

Step 3: Stable Local Extrema

A local extrema is observed if its derivative in scale space is stable and if it is on an apparent edge. If an extremum is decided as unstable or is placed on an edge, it is removed because it cannot be reliably detected again with small variation of viewpoint or lighting changes.
Step 4: Assigning Descriptor

A 16x16 window is used to generate a histogram of gradient orientation around each local extremum. To make the descriptor orientation invariant, all gradient orientations are rotated with respect to the major orientation of the local extremum.

c) Dimensionality Reduction Using PCA

Dimensionality reduction is performed to avoid the complexity and degradation introduced by the phenomenon called “Curse of Dimensionality”. A Dimensionality reduction algorithm aims to reduce the dimension by retaining only those data that are most relevant for the recognition task.

Principal Component Analysis (PCA) (Joliffe, 1986) is a standard technique for dimensionality reduction and has been applied to a broad class of computer vision problems, including feature selection, object recognition and face recognition. As the number of SIFT features is very high, the aim of using PCA in this study, is to reduce the number of SIFT features. Unnecessary features sometimes are strongly correlated with other features or constants which can be removed without affecting the performance of the recognizer. Principle component analysis is an unsupervised approach to extract the appropriate features from the data.

To achieve this, first the D-dimensional mean vector $\mu$ and the $D \times D$ covariance matrix $\Sigma$ are computed from the full data set. Then the eigenvectors and the eigenvalues are computed. The eigenvalues and eigenvectors are sorted according to decreasing absolute eigenvalue. Let $v_1 \ldots v_D$ be the eigenvector with eigenvalue $\lambda_1 \ldots \lambda_D$ respectively. Then the k eigenvectors with largest eigenvalues are chosen to form a $k \times D$ matrix $A$. Using this matrix, the data vectors are projected to a $k$-dimensional subspace.

$$X' = A(x - \mu)$$

(3.29)
This method can be applied to any type of data and this study uses it to reduce the dimensionality of the SIFT features.

### 3.6.5. Feature Fusion

The two feature vectors, LBP around pores and SIFT are combined to form a new fused feature vector to combine the advantage of both the techniques. The technique used for this purpose is the weighted sum rule and the process is explained below.

Let \( L = (L_1, \ldots, L_d) \) and \( S = (S_1, \ldots, S_d) \) be the LBP around pores and SIFT feature vectors. The fusion of \( L \) and \( S \) using weighted sum rule is done using Equation (3.30).

\[
\xi = (l_1 + \theta s_1, \ldots, l_d + \theta s_d) \ldots
\]  

(3.30)

As the dimension \( (d) \) of \( L \) and \( S \) might vary, a simple technique of padding the small dimension vector with zero was utilized. For example, let \( l \) be the vector with small dimensionality, the fusion feature is denoted as

\[
f_i = (s_i^{1} + \theta l_i^{1}, \ldots, s_i^{m} + \theta l_i^{m}, s_i^{m-1} + \theta l_i^{m-1}, s_i^{m} + \theta 0^0, \ldots)
\]

The value of \( \theta \) was set to 0.43 during experimentation. This value was obtained through multiple runs with different weight values.

### 3.7. MATCHING PROCESS

This section presents the two selected matching algorithms for recognizing partial fingerprints.

#### 3.7.1. Score based Matching

The algorithm for matching a partial and full image pair is based on distance between two LBP feature histograms. Minimum distance corresponds to best match. To get distance between two histograms, chi-square formula is used. Distance between two histograms \( S, M \) can be defined.
where \( n \) is the number of elements in the histogram. Chi-square formula is an effective measurement of similarity between two histogram, hence it is suitable for pair of nearest neighbour. For identifying a partial image in the set of full images, match score corresponding to each (partial image, full image) pair is obtained. If this difference is less than a threshold, then it is considered as a match and the score is updated accordingly. When the score reaches a maximum level, then the fingerprint is declared as a match. The score based matching using LBP around pore features consists of the steps given in Figure 3.32.

- Determine distance between two LBP feature histograms using Chi-Square measurement.
  - Let \( P_p \) and \( F_p \) be two histograms for pores in partial and full fingerprint image respectively
  - Calculate match score for each pair (partial and full fingerprint) using the following matching procedure

\[
\text{for } p \text{ in } P_p \\
\text{dis},i = \min(\text{chi-square-distance}(p, F_p,i)) \\
\text{if } \text{dis} < \text{threshold} \\
p \text{ is matched with distance } \text{dis} \\
F_p = F_p - F_p,i, \\
\text{else} \\
p \text{ is non-matched}
\]

- The full fingerprint with maximum match score is identified as the best match

**Figure 3.32 : Score based Matching using LBP around Pore Features**

The score based matching using SIFT features consists of the steps given in Figure 3.33.
While using SIFT features, an additional step of arranging the feature points of partial fingerprints in ascending order of closest neighbours is performed initially and the distance is calculated between the corresponding features of the full fingerprint. If the distance is greater than a threshold, it is said to be a match. When the total score exceeds a threshold value, it is considered to be recognized.

### 3.7.2. Neural Network based Matching

The neural networks are quite famous to be well adapted for problems of classification, where the task of recognizing the fingerprints as genuine and forgery was addressed as a two-class classification problem, in a given feature space. This approach consists of training a classifier (neural networks) on a training set made up of labeled Genuine and Imposter, to find a decision function in the considered feature space. Such decision function is then used at operation phase to label new images. In the present research work, a supervised Back Propagation Neural Network is used to classify the images as either recognized (genuine) or not-recognized (forgery).
Back-propagation (BP) neural network was invented by David Parker, Yaun LeCun, David Rumelhart, Geoffrey Hinton and Ronald William in the mid 1980s. It is a very popular learning algorithm to train multilayer networks with differentiable transfer functions to perform function approximation, pattern association and classification. The algorithm is named for how it handles errors in the network. The derivatives of the network error with respect to the network bias and weights are computed starting from the very last layer up to the first layer in order to modify the weights and biases of the network. The process starts from the output layer and goes back to the first layer and is therefore called the “Back-Propagation Network”.

Typically, a back-propagation network has an input layer, one or two hidden layers and one output layer. Although there is no limit on the number of hidden layers, generally one or two hidden layered networks are selected due to the complexity of the resulting system.

In the present research work, a Back Propagation Neural Network (BPNN) is used in the design the partial fingerprint recognition models. Neural networks have the ability to learn complex data structures and BPNN is preferred as it refers to the fact that any mistakes made by the network during training get sent backwards through it in an attempt to correct and teach the network to recognize genuine fingerprints. The algorithm is presented in Figure 3.34.

3.7.3. Back Propagation Neural Network Architecture

This research work is implemented using MATLAB to construct Back Propagation Neural Network (BPNN) that classifies partial fingerprints as genuine or not-genuine. The following method is used during matching based on BPNN.
The features extracted from the images are transformed into a vector having ‘n’ columns, where n is the number of dimensions. The target will be a vector of the same length as the number of inputs consisting of +1 for the genuine fingerprints and -1 for the imposter images. For example if the first three are versions of the reference image and the next 10 are imposter images then the target vector will be [1,1,1,-1,-1,-1,-1,-1,-1,-1,-1]. The network tries to map (generate weights at each connection neuron) the target vectors to the input vectors. Once it is trained, when presented with a test partial fingerprint image, the BPNN responds with a number close to 1 if the image matches the reference image or a number close to -1 if the image does not match.

The BPNN is constructed as a 3-layer network with input, hidden and output layers. The proposed architecture consists of one input layer, one output layer and one hidden layer. The network architecture is shown in Figure 3.34: BPNN Algorithm.

- Initialize the weights to small random values and choose a positive constant c.
- Repeatedly set equal to the features of images 1 to N, cycling back to sample 1 after sample N is reached.
- Feed forward step. For k=0,..., k−1, compute for all node.
  \[
  x_j^{(k+L)} = Re\left(\sum_{i=0}^{M_k} w_{ij}^{(k+1)} x_i^{(k)}\right)
  \]
- The sigmoid threshold function \( R(s) = \frac{1}{1+e^{-s}} \) is used.
- Calculate \( \delta_j^{(k)} = x_j^{(k)}(x_j^{(k)} - d_j) \)
- For layers k=K-1,...,1 compute
  \[
  \delta_j^{(k)} = x_j^{(k)}(1-x_j^{(k)})\sum_{i=0}^{M_{k+1}} w_{ij}^{(k+1)} x_i^{(k+1)}
  \]
- Replace \( w_{ij}^{(k)} \) by \( w_{ij}^{(k)} - cx_{ij}^{(k-1)} \delta_j^{(k)} \)
- Repeat until weights increase to change significantly.

**Figure 3.34 : BPNN Algorithm**
layer and two hidden layers. The weight of a neuron from its previous layer, generates a signal with the help of the activation function, \( \varphi(\sigma) = 1/(1+e^{-\sigma}) \), and transfers the signal to the next layer. BPNN architecture is shown in Figure 3.35.

![BPNN architecture](image)

**Figure 3.35 : BPNN architecture**

Choosing the optimum number of hidden layers and nodes will improve the training efficiency. The layers and nodes selected are based on trial and error selection method. The input node depends on the features extracted from the fingerprint images. The hidden layer consists of 1000 neurons and output layer consists of a single neuron. The activation function used in this research is tan sigmoid function. The learning algorithm employed in this study uses the Levenberg Marquardt (LM) method for minimizing the training error, which is defined as mean square difference between the desired output and the actual output. Network parameters are defined as follows:

\[
\begin{align*}
\text{net.trainParam.epochs} &= 500 \\
\text{net.trainParam.goal} &= 1e-5 \\
\text{net.trainParam.show} &= 50 \\
\text{net.trainParam.lr} &= 0.006 \\
\text{net.trainParam.mu} &= 0.6
\end{align*}
\]
The network is trained ten times using the tan sigmoid activation function and the average result is considered. The model is tested by using the standard rule of 60/40, where 60% of the samples are used for training and 40% is used for testing. In this classification method, training process is considered to be successful when the MSE reaches the value 1e-005.

### 3.8. PERFORMANCE EVALUATION

The method using evaluation of the proposed methods in each phase is given in Figure 3.36. Four stages of performance evaluation is planned are evaluate the techniques proposed in four phases of the study. The results are compared with their traditional counterparts to analyze the efficiency gained due to enhancement. The first stage of experiments are planned to evaluate the preprocessing algorithms in terms of Peak Signal to Noise Ratio, Figure of Merit, Mean Structural Similarity Index and speed of enhancement. The second stage of experiments evaluates the segmentation process to analyze the efficiency of the proposed segmentation algorithms. The performance is evaluated in terms of stability criteria, anti-noise criteria and segmentation speed.

![Figure 3.36 : Evaluation Methods](image)

Data Set

- Preprocessing
- Segmentation
- Feature Extraction
- Recognition

Base Method

Proposed Method

Performance Evaluation and Comparison
The third stage of experiments is used to evaluate the feature extraction process in terms of the number of features selected. The final stage of experiments are planned to analyze the performance of the proposed six partial fingerprint recognition systems. Experiments are also performed to analyze the effect of preprocessing and segmentation on fingerprint recognition accuracy. The six models are analyzed using False Acceptance Rate (FAR), False Rejection Rate (FRR), Equal Error Rate (EER), Total Error Rate (TER), Overall Accuracy and Execution Time.

3.9. PROPOSED RECOGNITION MODELS

The study builds different recognizers by applying the three feature sets created in Section 3.6 and two matching algorithms discussed in Section 3.7. The preprocessing (Section 3.4) and segmentation (Section 3.5) steps are kept as optional steps. While using the machine learning classifier (BPNN) and SIFT feature set, a process which reduces the dimensionality of the feature vector is performed. The dimensionality reduction in the present work is performed by PCA. The method used for building the Automatic Partial Fingerprint Recognition System is shown in Figure 3.37.

Using the three feature vectors (LBP around Pores, SIFT and Fusion features) and two matching algorithms (Score based and BPNN based), six recognizers are proposed for partial fingerprint recognition, grouped into two categories.

1. **Single-feature based Recognizer**: In this method, the matching algorithm compares a single feature (LBP-based pore feature vector or SIFT feature vector) against the template using a threshold to find a successful identification of a fingerprint. The matching algorithm is either score based or BPNN based.

2. **Fusion-feature based Recognizer**: Here, the two feature vectors (LBP around pores and SIFT) are combined using a weighted sum rule
method and are used during comparison. The matching algorithm is either score-based or BPNN-based.

The six proposed models are listed below.

1. Score based Matching using LBP around Pores
2. NN based Matching using LBP around Pores
3. SIFT and Score based Matching using SIFT
4. NN based Matching using SIFT
5. Score based Matching using Fused Features
6. NN based Matching using Fused Features
The proposed matching methods are evaluated using various performance metrics (section 3.8) to study the strengths of each recognizer and find the winning model that best suits partial fingerprint recognition.

3.10. CHAPTER SUMMARY

The main focus of the present research work is to develop partial fingerprint recognizers that can identify genuine and forgery fingerprints efficiently using pores and SIFT based features using score-based and BPNN. Six models are proposed for this purpose. Various experiments were conducted to test and evaluate the performance of the proposed models. The results obtained are presented in the next chapter, Results and Discussion.