Automated Vehicle License Plate Detection System Using FRIT Algorithm

by

Faruq Ahmed Jewel

MASTER OF SCIENCE IN INFORMATION AND COMMUNICATION TECHNOLOGY

Institute of Information and Communication Technology

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

2013

The thesis titled "**Automated Vehicle License Plate Detection System Using FRIT Algorithm**" submitted by Faruq Ahmed Jewel, Roll no: M10063103P, Session October 2006 has been accepted as satisfactory in fulfillment of the requirement for the degree of Master of Science in Information and Communication Technology on 31st December, 2013.

BOARD OF EXAMINERS

1. Dr. MD. Liakot Ali — Chairman (Supervisor)
   Professor
   Institute of Information and Communication Technology
   BUET, Dhaka-1000.

2. Dr. Md. Saiful Islam — Member
   Professor and Director
   Institute of Information and Communication Technology
   BUET, Dhaka-1000.

3. Dr. Md. Saiful Islam — Member (Ex-Officio)
   Professor and Director
   Institute of Information and Communication Technology
   BUET, Dhaka-1000.

4. Dr. Khosru M. Salim — Member (External)
   Associate Professor
   School of Engineering & Computer Science,
   Independent University, Bangladesh.
   Baridhara, Dhaka-1212.

CANDIDATE'S DECLARATION

It is hereby declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

_____

Faruq Ahmed Jewel

DEDICATED TO MY DAUGHTER "TASNIA AHMED NILOM"

Acknowledgement

Foremost, I would like to express my sincere gratitude to my supervisor, Dr. Md. Liakot Ali for the continuous support of my Masters Study and research from my heart, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. This is my realization that, I could not have imagined having a better advisor and mentor for my Masters study and complete the research work.

I also really thankful to Maruf Ahmed and Md. Rezaur Rahman my colleagues, for being with my side from start to end of this thesis.

My sincere thanks also goes to IICT Office staffs for provide logistic support to me to successfully complete the thesis work.

Last but not the least, I would like to thank my family: my parents Md. Arshad Mia and Nilima Akter, for giving birth to me at the first place and supporting me spiritually throughout my life and My Wife Maksuda Begum for mental support.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the project.

-Faruq Ahmed Jewel

Table of Contents

Abstract

Automatic vehicle number plate recognition (AVNPR) is an important research area in intelligent transportation system and it plays an important role in numerous applications. In AVNPR system license plate detection from the original image is the most critical part. It is quite challenging due to the diversity of plate formats and the non-uniform outdoor illumination conditions during image acquisition. Image transformation technique is widely used for license plate location and detection. A number of techniques are proposed in the literature but they are having accuracy problem while plate size is minimum, computationally very expensive and slow. In this thesis a novel technique has been developed for the license plate detection system using Finite Ridgelet Transform (FRIT) algorithm. The proposed technique is simple and effective which outperforms other existing methods for License Plate Detection (LPD). To verify the effectiveness of the proposed technique, experiments have been conducted using benchmark vehicle images. This gives remarkable success of detection in real time processing. Experiments result shows that the proposed technique is capable of detecting license plate successfully in various lighting condition. This technique is completely language, font, color and size independent. It is rotation invariant and capable of detecting license irrespective of poor illumination condition, multiple license plates in a same image and low quality noisy image. It is also effective in the shadow condition or the plate image is blurred. Achieved from the proposed technique the results have been compared with those off other research and shows that our technique. We also compare the results and efficiency of this work with other existing systems/techniques which gives better result.

In this system first an image is captured by a capturing device and then the process of license plate detection method is performed. After detecting the license plate the segmented image is stored in a database. Then from the database the process of character segmentation is performed. The processed information then can be used for several applications and can be controlled from a central control room.

Computer vision and character recognition algorithms for License Plate Recognition (LPR) are used as core modules for Automatic Vehicle Number plate Recognition (AVNPR) system [**2**]. An AVNPR system consists of three main parts: license plate detection (LPD), character segmentation (CS), and character recognition (CR). Among these three parts, the LPD is the most important stage and also the most difficult part [**2**]. This is mostly because during this stage we need to overcome various undesired input image conditions such as out of focus (blur) images, undesired illumination conditions, small size plates, rotations, shadows, and different weather conditions. Different LPD techniques are introduced in the literatures. Although the detection accuracy from those research is in the acceptable range but they are burdened with computational complexities the research work presented in this thesis focuses to overcome the limitations of the existing research on LPD and find a solution for the problem. In this thesis, a novel technique has been developed using FRIT for image representation algorithm for license plate detection which shows that it is capable of handling all the challenging scenarios in LPD and the computation speed is also real time.

1.2 Review of Previous Works and Observation

LPD is one of the important components in the Intelligent Road Transportation System and it has a lot of practical applications such as in automatic toll collections, parking fee payment, detection of vehicle crossing speed limits and thereby reducing road accidents etc. Due to these demanding applications for modern intelligent road transportation system, LPD is a challenging research area to the researchers all over the world since the last decade. Literature [**1**] presents a feature-based license plate localization algorithm that copes with multi-object problem in different image capturing conditions. It extracts

license plate candidates using edge statistics, morphological operations, color analysis and removes the incorrect candidates according to the determined features of license plates. But this solution does not provide a high degree of accuracy in natural scenery, since color is not stable when the lighting conditions change. In addition, as these methods are color based, they are country specific.

In the literatures [**2-3**] VLPD is based on edge statistics analysis. The authors used the statistical information of license plate regions based on the license plate edge analysis. This process is good for LPD but they are not robust in term of real-time detection. All the time the license plate may not available in the selected regions, then finding the edges in that region take long time but result become zero; Therefore these proposed methods are not suitable for real-time AVNPR systems.

The proposed approach in the literature [**4**] can only recognize license plate recognition systems when the plate orientation is straight to the camera and also the proposed system can tolerate slight tilting of the license plate, but the reality is that all the time it is not easy to get the right oriented license plate. So this approach is not applicable for the real time license plate detection.

Literatures [**5-6**] use morphological filtering as their base method. Morphological operations are inexpensive and robust, but this method suffers from various image conditions such as, illumination condition, blurriness of images, skew conditions, etc. which in-turn decrease the efficiency of detection (maximum success rate is 97.86%). So the approach is not good for real-time AVNPR system.

Image transformation method based on Hough and Radon transformations (HT and RT) for VLPD is proposed on the literatures [**7-8**]. In this method, edges in the input image are detected first. Then, HT is applied to detect the LP regions. However the authors acknowledge that the execution time of the HT requires too much due to computational complexities when applied to a binary image with great number of pixels.

In the literatures [**9-11**] authors proposed VLPD methods based on neural network and combination of plate features. But all these proposed techniques suffer from the trade-off between efficiency and effectiveness in terms of success rate and computational cost.

In this literature [1**2-13**] presents the method of detection of license plate from an image based on Maximally Stable Extremal Region (MSER). This is a good process when the image size is small. The computational complexity is very high because of this algorithm is completely depends on the numbers of pixels. So this type of method is not suitable for real time license plate detection.

The proposed approach in the literature [**14**] is template matching which is good for still image, offline image processing, when image processing time is not concern. This type of method is not perfect for any real-time image processing.

In this literature [**15**] authors proposed Wavelets based method for license plate detection and this is good approach for still image. But problem is that wavelet alone is not enough for detecting license plate when any car is moving due to the algorithms complexity.

The proposed method in the literature [16-18] is AdaBoost (GAB) algorithm and vertical boundary pairs both have a higher detection rate and a lower false positive rate. But these types of algorithms are taken much time for detection due to the algorithmic complexity. These algorithms are not applicable for real time object detection.

From the above literatures we see that there occurs significant progress in the research in LPD, but the proposed researches in this area are typically restricted to well-defined working conditions to obtain predictable scene features such as fixed illumination, fixed color, fixed country, limited vehicle speed, designated routes, and stationary backgrounds. Accurately and efficiently detection of license plates from images is always challenging due to the following reasons: (**1**) the size, shape and position (rotation) of the plate may vary. (**2**) The illumination condition in the image may vary. (**3**) The plate may be of any color, any type and color of font and the background color may be very similar to that of the plate. (**4**) The image may contain a number of noises.

(**5**) The image may be blurred which is a common scenario for motion compensated pictures. (**6**) Obstacles such as shadows, dirt and dust may make the system too difficult to detect the license plate. Again increased mobility and internationalization set the challenge of developing an effective LPD system that could handle plates from various countries with different character sets and syntax. So there are still a lot of scopes in conducting research for developing robust, accurate and country independent LPD system.

1.3 Thesis Objectives

The aim of this project is to develop fast and accurate automated vehicle license plate detection system for real time application. To achieve this goal the thesis has the following objectives:

(i) To develop a novel technique for License Plate Detection (LPD) for an AVNPR system, which should full-fill the requirement of a real-time system and efficient in general sense.

(ii) To design and simulate the proposed LPD system

(iii) To verify the functionality and effectiveness of the technique in different scenarios using benchmark images of license plate.

1.4 Outline of the Thesis

The thesis paper is organized as follows:

In chapter 1, Overview of the license plate detection system has been described. In Chapter 2, concepts of the basics image processing are introduced. Chapter 2 also includes a review of Finite Ridgelet Transform (FRIT). In chapter 3, the process of License Plate Detection (LPD) system is discussed. Chapter 4 includes several tests and results of this newly developed system. We have compared our technique with others and showed the results. In this chapter we also showed several experiment results under various difficulties and conditions. Finally, conclusion of the research work is given in chapter 5.

Chapter 2

Fundamentals of License Plate Detection

2.1 Introduction

This chapter discusses the fundamentals of image processing, image processing techniques used in Automated Vehicle Number Plate Recognition System. A brief review of Finite Ridgelet Transform (FRIT) is also included.

2.2 Image Description

A digital image is a representation of a two-dimensional image using ones and zeros (binary). Depending on whether or not the image resolution is fixed or not, it may be of vector or raster type. The term "digital image" usually refers to raster images also called bitmap images. Raster images have a finite set of digital values, called picture elements or pixels. The digital image contains a fixed number of rows and columns of pixels. Pixels are the smallest individual element in an image, holding quantized values that represent the brightness of a given color at any specific point. Typically, the pixels are stored in computer memory as a raster image or raster map, a two-dimensional array of small integers. Raster images can be created by a variety of input devices and techniques, such as digital cameras, scanners, coordinate-measuring machines, seismographic profiling, airborne radar, etc. They can also be synthesized from arbitrary non-image data, such as mathematical functions or three-dimensional geometric models; the latter being a major sub-area of computer graphics.

2.3 Types of Image

There are different image representation techniques in computer graphics, a raster graphics image or bitmap is a data structure representing a generally rectangular grid of pixels, or points of color, viewable via a monitor, paper, or other display medium. Raster images are stored in image files with varying formats. A bitmap is technically characterized by the width and height of the image in pixels and by the number of bits

per pixel (a color depth, which determines the number of colors it can represent). Each pixel of a raster image is typically associated to a specific 'position' in some 2D region, and has a value consisting of one or more quantities (samples) related to that position. Digital images can be classified according to the number and nature of those samples: (i) Binary, (ii) Gray Scale, (iii) Color, (iv) False-color, (v) Multi-spectral, (vi) Thematic, (vii) Picture function.

A binary image is a digital image that has only two possible values for each pixel. Typically the two colors used for a binary image are black and white though any two colors can be used. The color used for the object(s) in the image is the foreground color while the rest of the image is the background color. Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit (0 or 1). The names black-and-white, B&W, monochrome or monochromatic are often used for this concept,

A gray-scale digital image is an image in which the value of each pixel is a single sample, it carries only intensity information. It is composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest [19]. Gray-scale images are distinct from one-bit black-and-white images, which in the context of computer imaging are images with only the two colors, black, and white (also called bi-level or binary images). Gray-scale images have many shades of gray in between. Gray-scale images are also called monochromatic, denoting the absence of any chromatic variation (i.e.: no color). Gray-scale images are often the result of measuring the intensity of light at each pixel in a single band of the electromagnetic spectrum (e.g. infrared, visible light, ultraviolet, etc.), and in such cases they are monochromatic proper when only a given frequency is captured. But also they can be synthesized from a full color image.

A (digital) color image is a digital image that includes color information for each pixel. For visually acceptable results, it is necessary (and almost sufficient) to provide three samples (color channels) for each pixel, which are interpreted as coordinates in some color space. The RGB color space is commonly used in computer displays, but other

2.4 RGB Color Model

Values of the Red, Green and Blue are between 0 - 255. Colors upgrade to darker towards to 0 and upgrade to lighter when towards to 255. This situation is explained in the Cartesian coordinate system. The (0, 0, 0) origin point is black, and (1, 1, 1) point is white. Any color occurs as a result of the merger red, green, blue color with certain coefficients in the coordinate system. Gray color is above the white and black level, combining diagonal corners**.**

2.5 Gray Level

The image is converted to gray level to accelerate the image processing. Thus, in the picture, there will remain only black, white and gray values. The process needs to be done before the image is converted into binary level. A general equation for converting a RGB color image into a Gray Level image is: **0.2989 * R + 0.5870 * G + 0.1140 * B,** where R, G, B correspond to current pixel's R, G, B values respectively.

2.6 Binary Level

An image consists of numeric values between 0 - 255. The numerical value of the picture is reduced to two values with binary level. Thus, an 8 - bit image is converted into 2 - bit format. The threshold value must be determined for this conversion. Using a fixed threshold value is not correct because of external factors such as sunlight, shadows at real-plate images. A distribution histogram is useful for calculating threshold value. If the pixel value in the image is greater than threshold value, then the pixel value is assumed as "0"; and if the image pixel' value is less then threshold value, the pixel value is assumed as "1". In this way the image is converted to the binary level.

2.7 Histogram / Histogram Distribution

A histogram uses a bar graph to profile the occurrences of each gray level present in an image **[20]**. Figure 2.2 shows a simple histogram. The horizontal axis is the gray-level values. It begins at zero and goes to the number of gray levels (256 in this example). Each vertical bar represents the number of times the corresponding gray level occurred

in the image. In Figure 2.2 the bars “peak" at about 70 and 110 indicating that these gray levels occur most frequently in the image. Among other uses, histograms can indicate whether or not an image was scanned properly. Histograms also help to select thresholds for object detection (an object being a house, road, or person). Objects in an image tend to have similar gray levels. For example, in an image of a brick house, all the bricks will usually have similar gray levels. All the roof shingles will share similar gray levels, but differ from that of bricks. In **Figure 2.2**, for example, the valleys between the peaks at about 60 and 190 might indicate that the image contains three major kinds of objects perhaps bricks, roof, and a small patch of sky. Practical object identification is never simply a matter of locating histogram peaks, but histograms have been important for the research in object identification **[20]**.

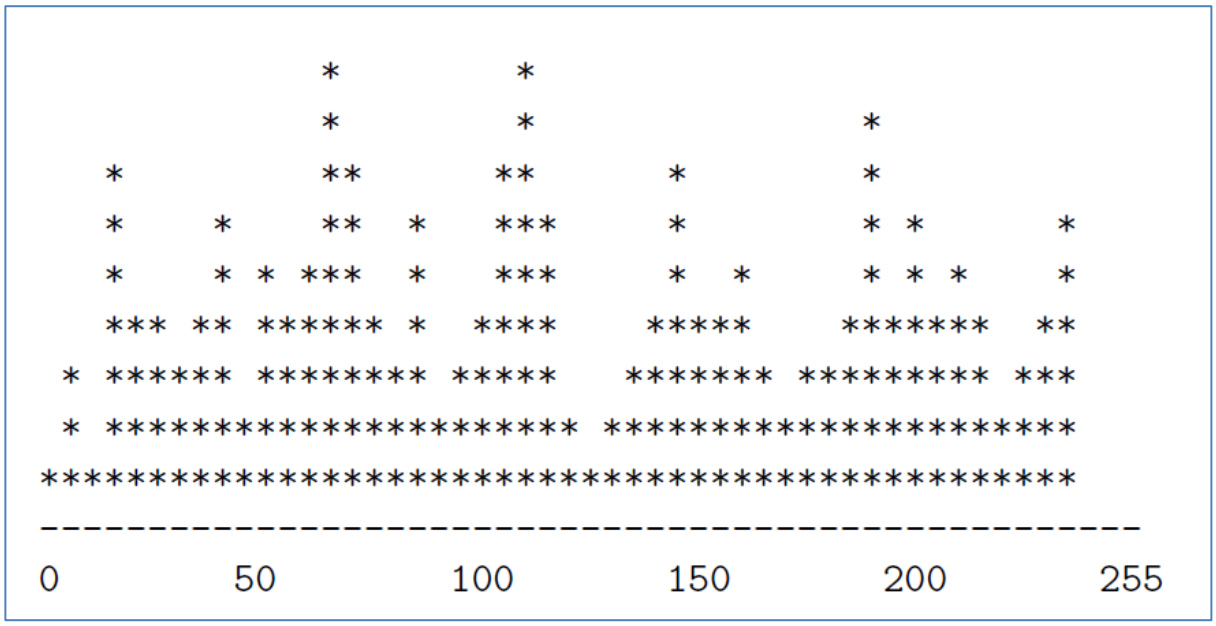


Figure 2.2: Simple Histogram

2.8 Histogram Equalization

Equalization causes a histogram with a mountain grouped closely together to “spread out" into a flat or equalized histogram. Spreading or flattening the histogram makes the dark pixels appear darker and the light pixels appear lighter. Histogram equalization

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	200	200	200	200	0	0	0
0	0	200	200	200	200	200	200	0	0
0	0	200	200	200	200	200	200	0	0
0	0	200	200	200	200	200	200	0	0
0	0	200	200	200	200	200	200	0	0
0	0	0	200	200	200	200	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 2.3.2: The Result of Eroding Figure 2.3.1

Figure 2.3: The Result of Erosion. Where Figure 2.3.1 is a binary image and applying the erosion technique over the binary image we have found the figure 2.3.2 which is smaller than the real binary image.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	200	200	200	200	200	200	0	0
0	0	200	200	200	200	200	200	0	0
0	0	200	200	200	200	200	200	0	0
0	0	200	200	200	200	200	200	0	0
0	0	200	200	200	200	200	200	0	0
0	0	200	200	200	200	200	200	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 2.4.1: A Binary Image

0	0	0	0	0	0	0	0	0	0
0	***	***	***	***	***	***	***	***	0
0	***	200	200	200	200	200	200	***	0
0	***	200	200	200	200	200	200	***	0
0	***	200	200	200	200	200	200	***	0
0	***	200	200	200	200	200	200	***	0
0	***	200	200	200	200	200	200	***	0
0	***	200	200	200	200	200	200	***	0
0	***	***	***	***	***	***	***	***	0
0	0	0	0	0	0	0	0	0	0

Figure 2.4.2: The Result of Dilating Figure 2.4.1

Figure 2.4: The Result of Dilation. Where Figure 2.4.1 is a binary image and applying the dilation technique over the binary image we have found the Figure 2.4.2 which is larger than the real binary image. There are two general techniques for erosion and dilation. One technique employs a threshold and the other technique uses masks to erode and dilate in desired directions.

The threshold technique looks at the neighbors of a pixel and changes its state if the number of differing neighbors exceeds a threshold. Figure 2.3.2 and Figure 2.4.2 used a threshold parameter of three.

The masking technique lays an **NxN** (3x3, 5x5, etc.) array of 1s and 0s on top of an input image and erodes or dilates the input. With masks we can control the direction of erosion ordination. Figure 2.5 shows four 3x3 masks (5x5, 7x7, etc. masks are other possibilities). The first two masks modify the input image in the vertical or horizontal directions while the second two perform in both directions.

It can be seen that two very bright spots are found in the Radon transform, and the positions show the parameters of the lines in the original image. A simple thresholding algorithm could then be used to pick out the line parameters, and given that the transform is linear many lines will just give rise to a a set of distinct point in the Radon domain.

Let us denote Z_p= {0,1,…, p −1} , Z_p^* = {0,1,…, p −1, p } , where p is a prime number. Note that Z_p is a finite field with modulo p operations. The **FRAT** [**23**] of a real function (image) f, on the finite grid $Z_p^2$is defined as,

$$\boldsymbol{r_{k,l}} = \frac{\mathbf{1}}{\sqrt{\boldsymbol{p}}}\sum_{(\boldsymbol{i},\boldsymbol{j})\in \boldsymbol{L_{k,l}}} \boldsymbol{f}[\boldsymbol{i},\boldsymbol{j}] = \frac{\mathbf{1}}{\sqrt{\boldsymbol{p}}}\sum_{\boldsymbol{i}=\mathbf{0}}^{\boldsymbol{p}-\mathbf{1}}\sum_{\boldsymbol{j}=\mathbf{0}}^{\boldsymbol{p}-\mathbf{1}} \boldsymbol{f}[\boldsymbol{i},\boldsymbol{j}]\boldsymbol{\delta}_{\boldsymbol{L_{k,l}}} = < f, \boldsymbol{\varphi_{k,l}} >, k \in \boldsymbol{Z_p^*}, \boldsymbol{l} \in \boldsymbol{Z_p} \quad \text{---------- (2.1)}$$

where $L_{k,l}$denotes FRAT lines on Z_p^2:

$$\begin{cases} L_{k,l} = \{(i,j): j = ki + l(mod\ p), i \in Z_p\}, \ 0 \le k \le p \\ L_{p,l} = \{(i,j): j \in Z_p\}; \end{cases}$$

$\varphi_{k,l}$denotes the basis on Z_p^2 corresponding to $L_{k,l}$: $\delta_{L_{k,l}}[i,j] = \begin{cases} 1, [i,j] \in L_{k,l} \\ 0, [i,j] \notin L_{k,l} \end{cases}$, $\varphi_{k,l} = \frac{1}{\sqrt{p}}\delta_{L_{k,l}}$

From the definition of FRAT lines, one can easily obtain that slope k ($k = p$ corresponds to the infinite slope or vertical lines) and intercept l have finite possible values and all FRAT lines have a common length of p. FRAT maps an image of size $p \times p$ in the image domain to a coefficientmatrix of $p \times (p+1)$, where the k -th column represents the FRAT coefficient sequence of the corresponding slope. FRAT is extraordinary suitable for line singularity presentation for its good energy concentration property. It is an invertible transform that can lead to perfect reconstruction. The inverse finite Radon transform (IFRAT) can be obtained by the finite back-projection (FBP) operator. That is, for any coefficient matrix $r: \{r_{k,l}\}_{k\in Z_p^*, l\in Z_p}$ in the FRAT domain.

$$\boldsymbol{IFRAT_r}[\boldsymbol{i},\boldsymbol{j}] = \boldsymbol{FBP_r}[\boldsymbol{i},\boldsymbol{j}] = \frac{\mathbf{1}}{\sqrt{\boldsymbol{p}}}\sum_{(\boldsymbol{k},\boldsymbol{l})\in \boldsymbol{P_{i,j}}} \boldsymbol{r_{k,l}}, (\boldsymbol{i},\boldsymbol{j}) \in \boldsymbol{Z_p^2} \quad \text{---------- (2.2)}$$

where $P_{i,j}$ denotes the sets of indexes that all go through a point $(i,j) \in Z_p^2$ More specifically, we can write $P_{i,j} = \{(k,l): l = j - ki(modp), k \in Z_p\} \cup \{(p,i)\}$.

Note that the modulo operator in eq. (**2.1**) leads to periodic "wrap around" which causes FRAT lines to exhibit different appearances from natural lines. The "wrap around" effect disperses energy distribution of natural lines in FRAT domain, thus representation ability is weakened. The energy dispersion between FRAT coefficients caused by the "wrap around" phenomenon is diverse, depending upon the locations, lengths of line singularities in the image domain, etc. Elimination of the "wrap around" effect is among the primary open problems in FRAT-based applications [**25]**. Do[**23**] constructed an optimal ordering FRAT by selecting the optimal normal vector from the normal vector parameter sets which all correspond to the same slice inspired by the projection slice theorem. We assume the optimal ordering FRAT is adopted in this paper in the sequel, where k is regarded as an index in the set of optimal FRAT normal vectors rather than a slope value. It could reduce the "wrap around" effect in some extent, but reordering of intercepts may lead to energy dispersion in FRAT coefficients of smooth regions in images. Xiao and Li tried to avoid probable "wrap-around" of certain lines by extending the original image to a larger size **[26]** at the cost of high storage and computation complexity.

2.11 Finite Ridgelet Transform - FRIT

Transformation methods provide the possibility to investigate signal in alternative perspectives and have been widely used in many image processing tasks such as coding, compression, de-noising, etc. The basic idea is if most information is packed into a relatively small number of independent transformation coefficients, their reconstruction provides a good approximation of the original signal. Wavelet transform has been extensively discussed in image processing and gets adopted in the international still image compression standard JPEG 2000. Its success is mainly due to the perfection of mono-variant polynomial factorization theory in mathematical and the simplicity of Mallat's one-dimensional signal decomposition structure in signal analysis engineering. Wavelet is good at catching zero-dimensional or point singularities only. Two-

dimensional discrete wavelet transform (2D-DWT) generated by a tensor-product of two perpendicular 1D wavelets is sensitive to horizontal, vertical and diagonal directions only, thus encounters difficulty in effective image presentation. To overcome the weakness of wavelet in higher dimensions, Candes et al. **[21]** pioneered Ridgelet Transform for representation of liner singularities in images.

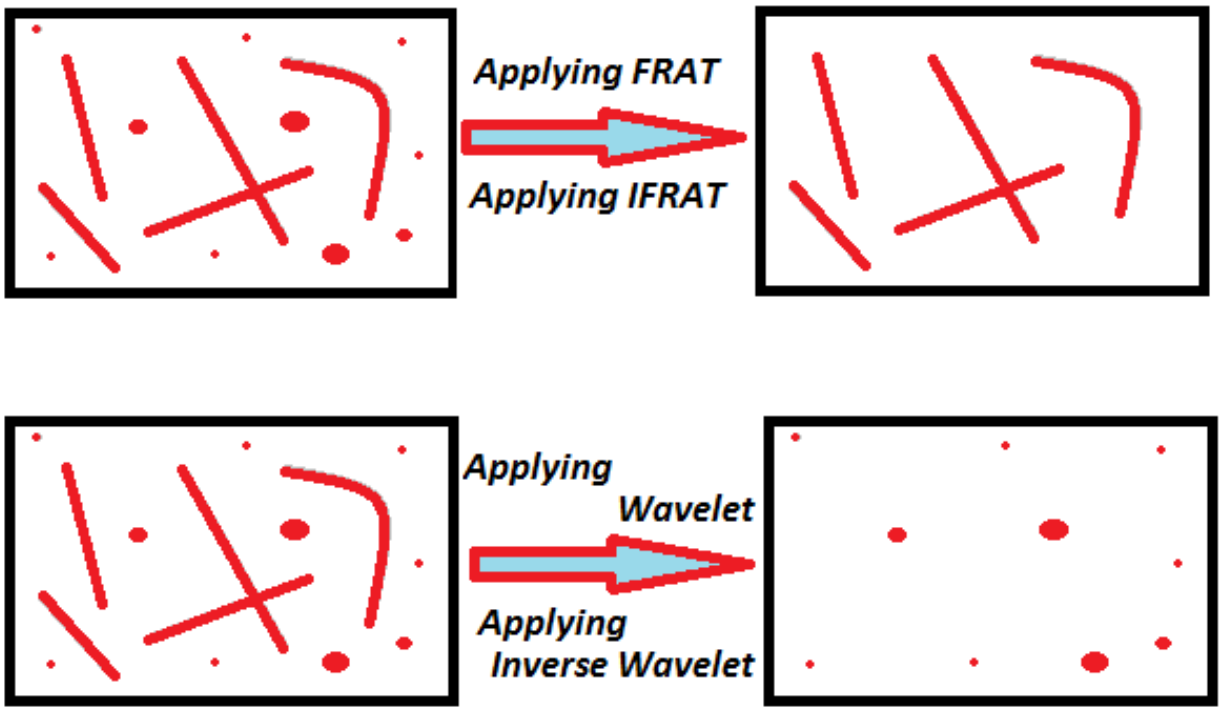


Figure 2.10: One dimensional wavelet transformation over FRAT domain is FRIT

The above image shows that finite radon transform only detect the line singularities and wavelet transformation only detect dot singularities. It can be seen that after applying FRAT and IFRAT all dots are removed from the images and only lines remain exist. In the same way applying wavelet and inverse wavelet transform all linear singularities are removed and only dots remain. From the definition of FRIT we know that applying wavelet transform over FRAT domain of an image is called FRIT transformation. FRAT transform converts all line into dots and wavelet work as a filter where it keeps all bright dots. So after applying IFRIT only represent the lines of original images. License plates are combination of line.

hand calculations. We shall illustrate many concepts by both simple hand calculations and more involved computer computations.

2.12.1 The Haar transform

In this section we shall introduce the basic notions connected with the Haar transform, which we shall examine in more detail in later sections. First, we need to define the type of signals that we shall be analyzing with the Haar transform.

A discrete signal is a function of time with values occurring at discrete instants. Generally we shall express a discrete signal in the form $f = (f1, f2, \ldots, fN)$, where N is a positive even integer which we shall refer to as the length of f. The values of f are the N real numbers $f1, f2, \ldots, fN$. These values are typically measured values of an analog signal g, measured at the time values t = t1, t2, . . . , tN. That is, the values of f are

$f1 = g(t1), f2 = g(t2), \ldots, fN = g(tN)$. - (2.3)

For simplicity, we shall assume that the increment of time that separates each pair of successive time values is always the same. We shall use the phrase equally spaced sample values, or just sample values, when the discrete signal has its values defined in this way. An important example of sample values is the set of data values stored in a computer audio file, such as a .wav file. Another example is the sound intensity values recorded on a compact disc. A non-audio example, where the analog signal g is not a sound signal, is a digitized electrocardiogram.

Like all wavelet transforms, the Haar transform decomposes a discrete signal into two sub signals of half its length. One sub signal is a running average or trend; the other sub signal is a running difference or fluctuation. Let's begin by examining the trend sub signal. The first trend sub signal, a1 = (a1, a2, . . . , aN/2), for the signal f is computed by taking a running average in the following way. Its first value, a1, is computed by taking the average of the first pair of values of f: (f1 + f2)/2, and then multiplying it by √2. That is, a1 = (f1+f2)/ √2. Similarly, its next value a2 is computed by taking the average of the next pair of values of f: (f3 +f4)/2, and then multiplying it by √2. That is,

a2 = (f3 + f4)/√2. Continuing in this way, all of the values of a1 are produced by taking averages of successive pairs of values off , and then multiplying these averages by √2. A precise formula for the values of a1 is

$$am = \frac{f2m-1+f2m}{\sqrt{2}} \quad \text{---------- (2.4)}$$

for m = 1, 2, 3, . . . , N/2. For example, suppose f is defined by eight values, say f = (4, 6, 10, 12, 8, 6, 5, 5); then its first trend sub signal is a1 = (5√2, 11√2, 7√2, 5√2). This result can be obtained using Formula (2.3).

2.13 Geometric Mean

In mathematics, the geometric mean is a type of mean or average, which indicates the central tendency or typical value of a set of numbers by using the product of their values (as opposed to the arithmetic mean which uses their sum). The geometric mean is defined as the nth root (where n is the count of numbers) of the product of the numbers.

For instance, the geometric mean of two numbers, say 2 and 8, is just the square root of their product; that is $\sqrt{2.8}$ =4.

A geometric mean is often used when comparing different items – finding a single "figure of merit" for these items – when each item has multiple properties that have different numeric ranges. For example, the geometric mean can give a meaningful "average" to compare two companies which are each rated at 0 to 5 for their environmental sustainability, and are rated at 0 to 100 for their financial viability. If an arithmetic mean was used instead of a geometric mean, the financial viability is given more weight because its numeric range is larger- so a small percentage change in the financial rating (e.g. going from 80 to 90) makes a much larger difference in the arithmetic mean than a large percentage change in environmental sustainability (e.g. going from 2 to 5). The use of a geometric mean "normalizes" the ranges being averaged, so that no range dominates the weighting, and a given percentage change in any of the properties has the same effect on the geometric mean. So, a 20% change in

2.14 Chapter Summary

In this chapter I have described the fundamentals of image type, processing, image processing techniques used in Automated Vehicle Number Plate Recognition System. Also discussed about Histogram Equalization and Morphological Operation. Also different types of Image transformation techniques like Finite Ridgelet Transform (FRIT),Finite Radon Transform (FRAT), Haar wavelet, Geometric mean, Which will help reader to understand next discussion clearly.

Chapter 3

License Plate Detection Systems

3.1 License Plate Detection

As earlier said, an LPR system consist of three main parts: license plate detection, character segmentation, and character recognition. Among these three parts the license plate detection (LPD) is the most challenging and crucial stage and also the most difficult part of an LPR system. This is mostly because during this stage we need to overcome various undesired input image conditions such as out of focus (blur) images, undesired illumination conditions, small size plates, rotations, shadows, and different weather conditions. The factors that influence the design of an LPR system include: i) Vehicle speed, ii) Volume of traffic flow, iii) Camera to license plate distance, iv) Ambient illumination, v) Types and variation of license plates, vi) Weather, etc. An automatic number plate recognition solution typically addresses four key issues: 1. Vehicle presence: Is a vehicle present?2. Plate location: Where is the number plate in the image?3. Glyph location: Where are the number plate glyphs within the plate?4. OCR (Optical Character Recognition): What are the characters on the plate?

Each of these issues can be addressed in many different ways, and some approaches may address more than one issue at once. In this paper, a novel technique using Finite Ridgelet Transform is presented. The proposed technique can detect plates of different sizes, different illumination conditions, rotations, scales, shadows, and the real world noise. Moreover this procedure was successful for many blurred images.

3.2 Principles of Number Plate Detection

The first step in a process of automatic vehicle number plate recognition (AVNPR) system is the detection of a license/number plate area. This problematic includes methods that are able to detect a tetra-lateral (as the license plate is not necessary to be a rectangle) area of the number plate in an original image. Humans define a number plate in a natural language as a “small plastic or metal plate attached to a vehicle for official

identification purposes", but machines do not understand this definition as well as they do not understand what "vehicle", "road", or whatever else is. Because of this, there is a need to find an alternative definition of a number plate based on descriptors that will be comprehensible for machines.

If we define a number plate by its color, a particular shape (eg. a 2:1 width-height ratio rectangle) or the text type it contains then most of the problems discussed earlier would arise and the technique would not be effective in a general sense. On the other hand if we define a license plate by its edge characteristics then we can bypass most of the issues related with the license plate detection technique. For example, we can think of a license plate by a region bounded by edges of a rectangular type of shape which contains a lot of horizontal and vertical edges because of the text inside it. If we think like this then we need not necessary to take concern of any other characteristics of a license plate. Moreover there is no impacts on the license plate detection where the license plate is rotated/skewed on the image, if we take concern of edge analysis.

So, we have defined the number plate as a **"rectangular area with increased occurrence of horizontal and vertical edges"**. The high density of horizontal and vertical edges on a small area in many cases caused by contrast characters of a number plate, but not in every case. This process can sometimes detect a wrong area that does not correspond to a number plate. Because of this, we may detect **several regions of interests (RoI)** for the plate by this technique, and then we choose the best one by a further analysis.

Let an input image to the system be defined by a function *f(x,y)* where *x* and *y* are spatial coordinates, and *f* is an intensity of light at that point. This function is always discrete on digital computers, such as$x \in N, y \in N$, where Ndenotes the set of natural numbersincluding zero. We define operations such as edge detection as mathematical transformations of function *f*. The detection of a license plate area consists of a series of convolution and modulo operations. Modified image is then thresholded by histogram thresholding method and then a series of morphological operations are performed for

acquiring the regions of interests. These RoIs are used to determine an area of a license plate.

3.3 License Plate Detection Using FRIT

Utilization of FRIT coefficients is based on the method of image de-noising using FRIT. The motivation for the FRIT-based image de-noising method is that in the FRIT domain, linear singularities of the image are represented by a few large coefficients, whereas randomly located noisy singularities are unlikely to produce significant coefficients. Therefore, a simple thresholding scheme for FRIT coefficients can be very effective in de-noising images that are piece-wise smooth away from singularities along straight edges [**23**]. Now our purpose is to detect the strong edges (not only the straight edges), as we want to detect the boundary edges of the license plate including the text edges on it. There is the freedom of manipulations FRIT domain according to the application requirements. Our target is to find bounded regions by edges containing several small size edges. In our case the image type is of natural scenery. In this type of images there a lot of edges including strong straight and curvy edges. If we could find object edges eliminating the background from image then we are one step ahead to detect license plate regions. Unfortunately an edge is not always a straight line. Because orthonormal ridgelet analysis amounts to a non-orthogonal wavelet analysis in Radon space in the two-dimensional case, an object with curved singularities is still a curvilinear one after Radon transform, not a point. Alternately, we can say that in analyzing an object which exhibits curved singularities, the ridgelet coefficients are not sparse, because the wavelet coefficients of Radon transform of the curved singularities are not sparse. For our purpose we need to find a threshold value. In this case we cannot use the universal threshold value given in [**23**] as this is for de-noising images and representing them. If we can find a threshold for which there are edges strongly in the image rather than the objects our purpose is served.

Several analytical analyses are performed on column and row wise FRIT coefficients (Figure 3.1(a, b, c, d) and Figure 3.2(a, b, c, d)) to find the threshold value, like their mean, variance, entropy, standard deviation. In many of these analyses, good results are

found by choosing a threshold value roughly greater than the geometric mean value of row and column wise FRIT coefficients. From a lot of trial and error it is found that the best result is achieved if we perform column wise scanning using the threshold value $\mathbf{T_c}$ and row wise scanning using the threshold value $\mathbf{T_r}$, where, $\mathbf{T_c}$= Column-wise geometric mean of FRIT coefficients, and $\mathbf{T_r}$= Row-wise geometric mean of FRIT coefficients.

So if we threshold the FRIT coefficients and discard the significant coefficients we can find an image where there are strong large edges and strong small edges but dispersed objects with edges. Thresholding the FRIT coefficients results an abstract image where the objects are not totally eliminated and the strong large and small edges are shown clearly. Therefore, the output image is basically an abstract image of the input image where the sharp objects are shown clearly with overall sharpness and image clearness decreased.. In Figure: 3.3 several results are shown using this thresholding method.

The type of thresholded output image is very useful to us, because when the image will be binarized by performing a histogram thresholding method, we shall acquire an image where the strong edges are shown by scattered pixels along their edges. This type of edges for LP can be overcome the problem of blurriness, rotation of LPs. Moreover it is not necessary to perform image enhancement operations (enhance brightness, contrast, reduce blurriness, etc.) before FRIT based thresholding technique, though it might increase the performance of the technique.

This thresholding method can be performed during the image transform and the process is fast since it consists of some logical operations When we are done with thisthresholding, we can start our procedure for LP detection. Since the processed transformed image is gray image, we have to convert it into a binary image for further operations. The next sections describe these procedures for LP detection.

3.3.1 Steps for FRIT Based Thresholding

In summary, for this thresholding all the significant FRIT coefficients are pulled to zero according to the thresholding values and others are remained intact. The outcome is an abstract image of the original input image where there are strong large and small edges. The necessary steps for performing the FRIT based thresholding is given below:

Step 1: Compute the column-wise geometric mean, $\mathbf{T_c}$ of the FRIT coefficients.

Step 2: Compute the row-wise geometric mean, $\mathbf{T_r}$ of FRIT coefficients.

Step 3: Threshold each original FRIT coefficients with $\mathbf{T_c}$ by performing a column-wise scanning of the FRIT coefficient array.

Step 4: Threshold each original FRIT coefficients with $\mathbf{T_r}$ by performing a row-wise scanning of the FRIT coefficient array.

Parameters for FRIT Thresholding Used in the Technique:

1) The input image size is 257x257, but the size can be extended to any NxN size image

2) We use Haar-Wavelet with DWT extension mode to zero padding.

3) The input image has to be normalized gray-level of double type.

3.4 Work Flow of the Proposed Technique

In this thesis, an LPD method is presented to overcome the issues raised in the literature, which include complex environment, rotation, lighting, low contrast, and blurriness. The proposed method also has high degree of freedom about license plate size and

orientation. Here we have declared the steps for the proposed technique for license plate detection using FRIT. Figure: 3.4below shows the work flow for the proposed method.

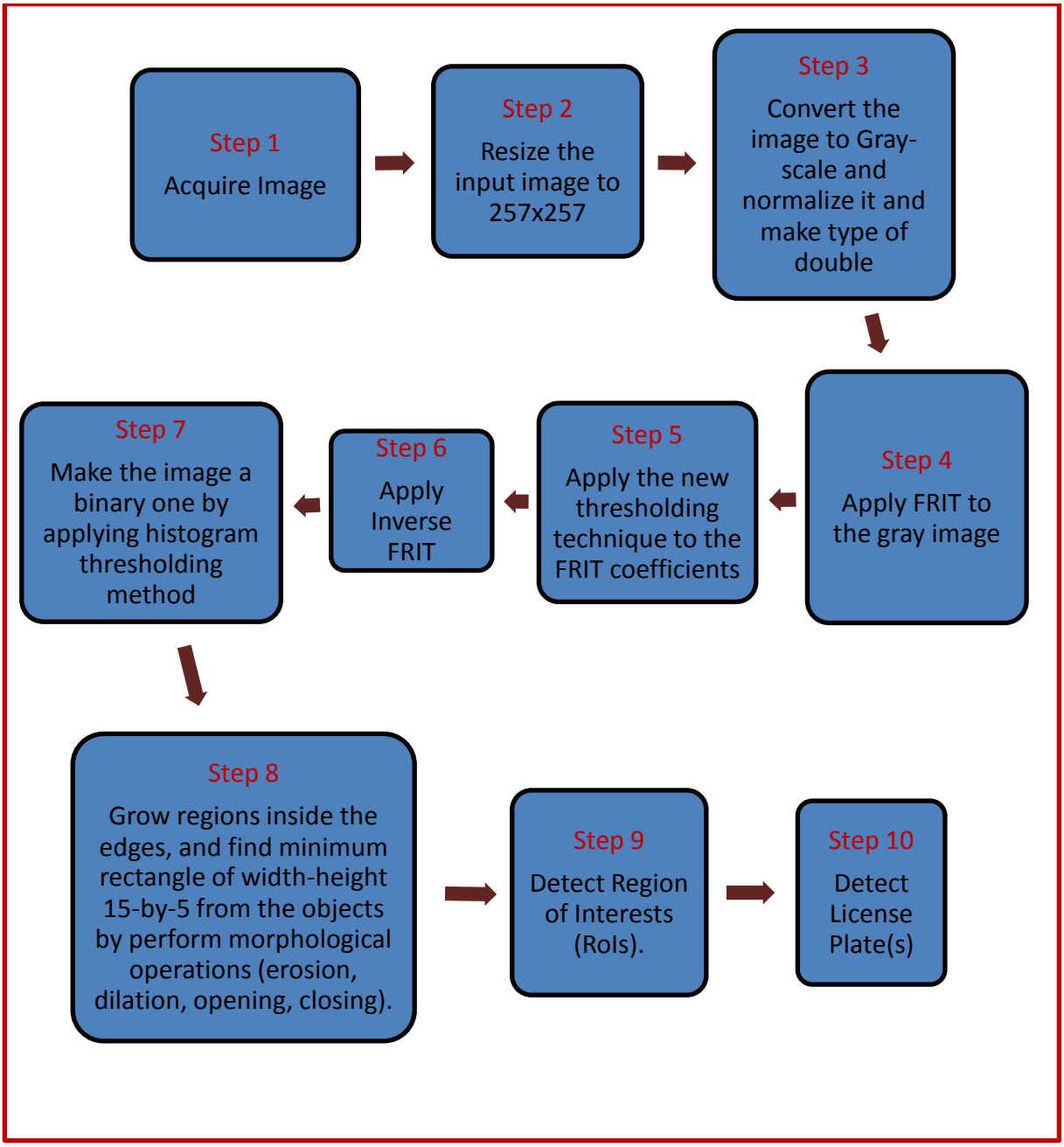
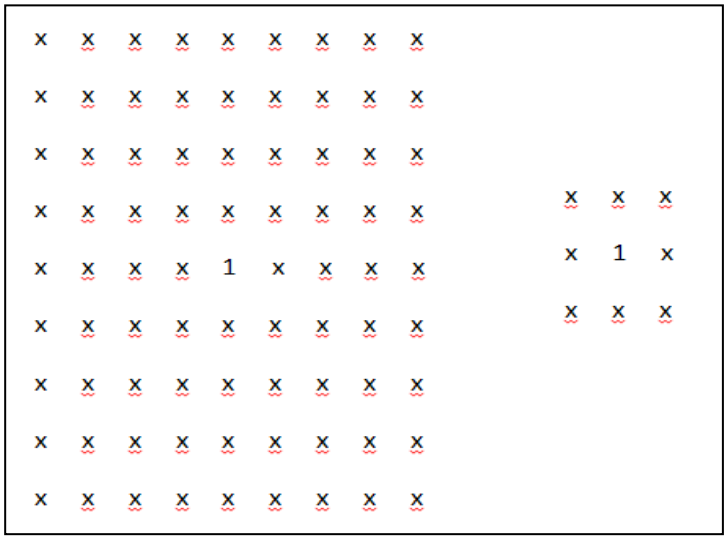


Figure: 3.4: Work Flow of the Proposed Technique

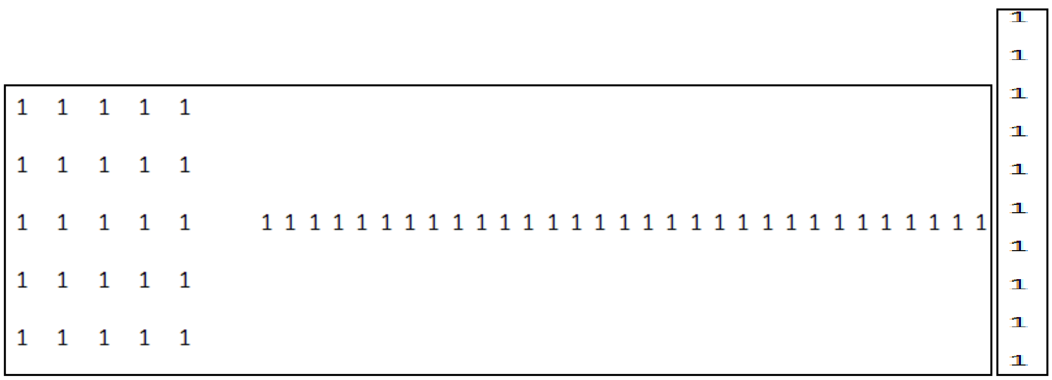
3.5 License Plate Localization and detection

This section provides methods performed to detect LP regions and verify them. A series of morphological operations are performed first to find the regions of interests. Then several verification methods are performed to detect the license plate correctly. The main theme is that after performing FRIT thresholding we convert the thresholded image into a binary image by histogram thresholding method, where the image is represented by scattered pixels closely together where the edges are. Next we erode this image by a thresholded erosion method so that the "salt and pepper" like pixels are gone. Then we perform dilation by a thresholded dilation method to connect the edges and grow solid region bounded by edges. In this stage we find several objects of various shapes. Now we discard the objects which are not in the shape of a tetra-lateral and are less than a width of 15 pixels and height of 5 pixels by performing a series of morphological erosions and dilations with various structuring elements(See Figure: 3.5below). Finally the output is an image with object(s) like tetra-laterals. This gives us the regions of interests (RoIs) (See Figure: 3.6 below). We can merge this image with the input gray level image or resized original image to verify the regions. Figure: 3.7shows the images of several sample images at several steps. The full MATLAB source code of the process is given in APENDIX.

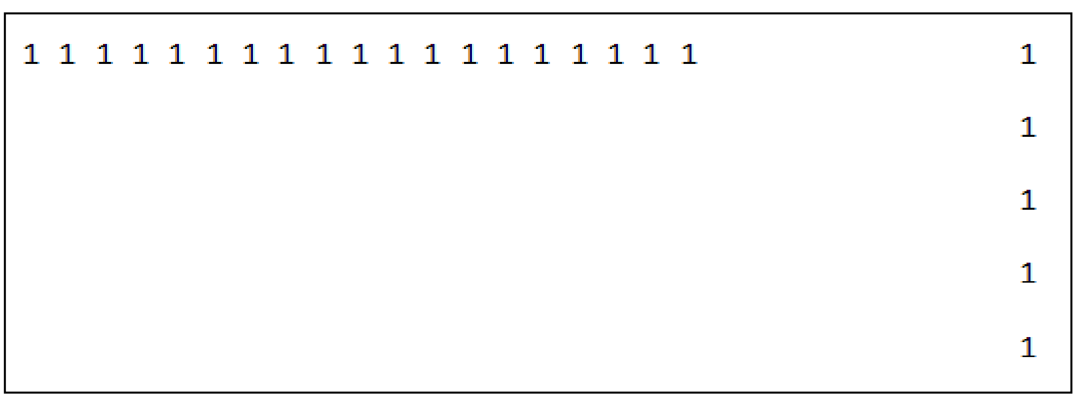


(a) Structuring elements for thresholded erosion

(b) Structuring elements for thresholded dilation



(c) Structuring elements for masked dilation



(d) Structuring elements for masked erosion

Figure 3.5:Various Structuring elements for performing morphological operations.

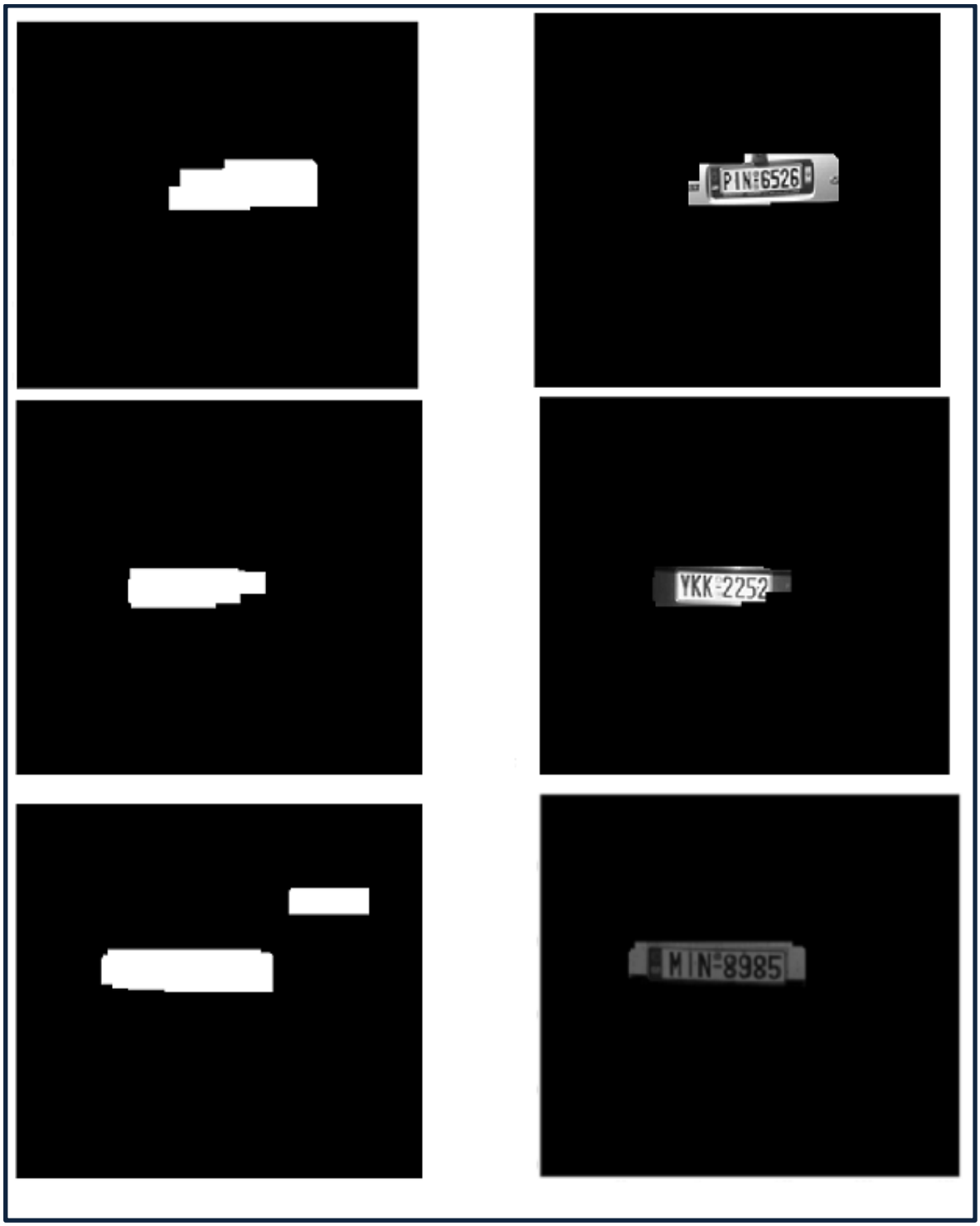
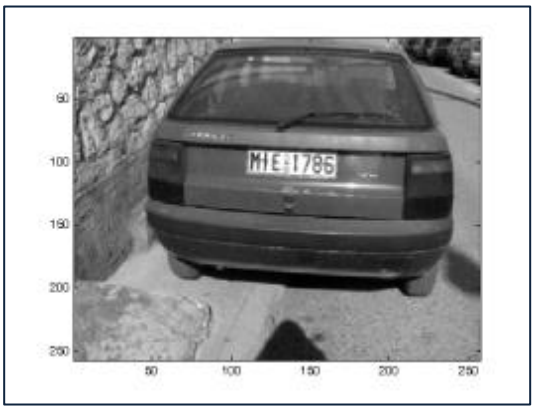
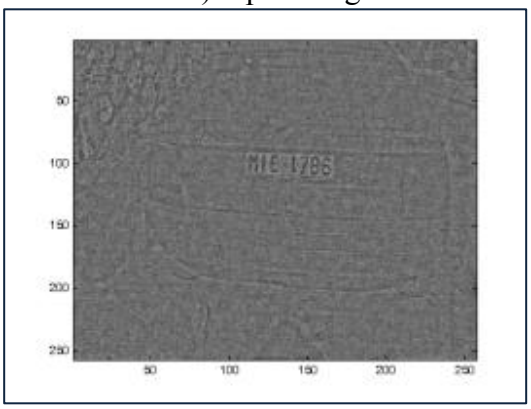


Figure: 3.6: Regions of interests at left and verified original input image license plates at right.

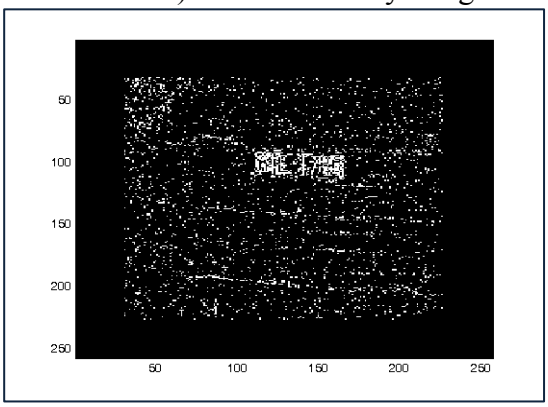
i) Input Image



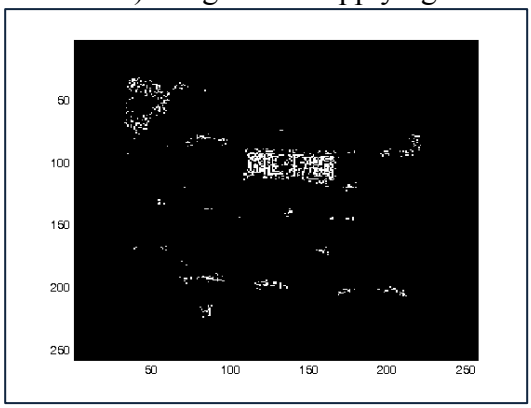
ii) Converted Gray Image

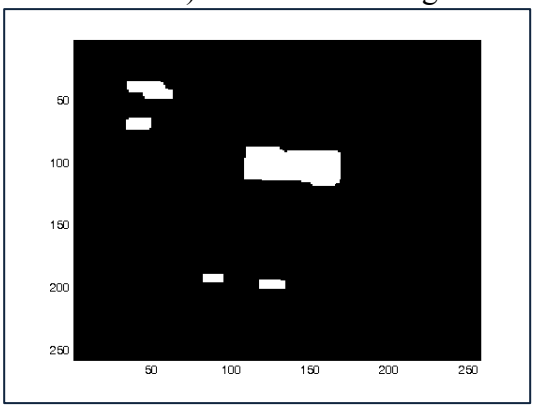


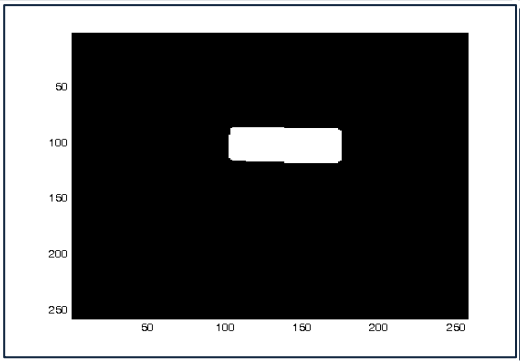
iii) Image After Applying FRIT

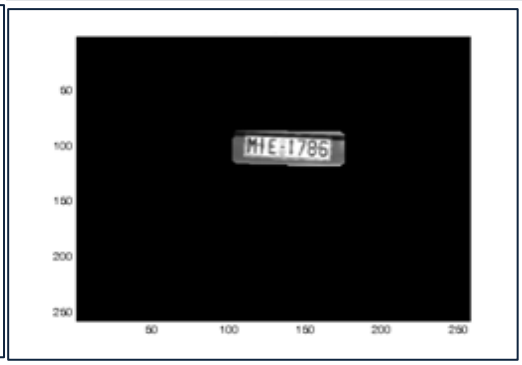


iv) Thresholded Image



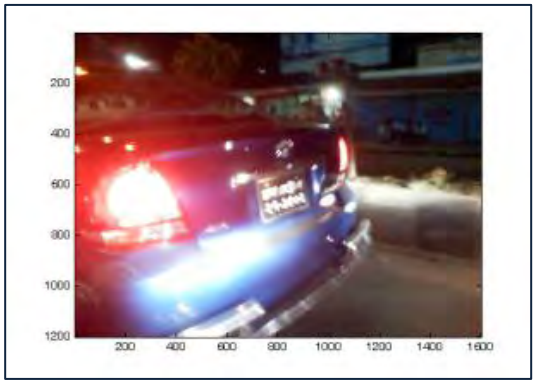




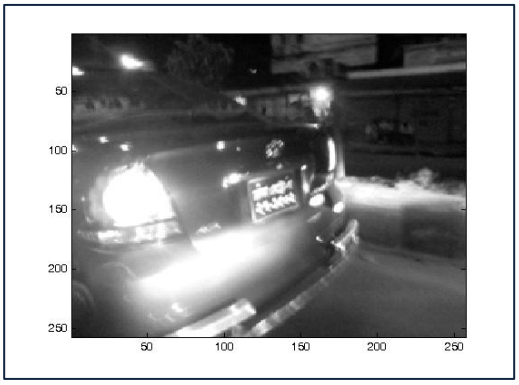


v) Image output after performing morphological operations and verifications.

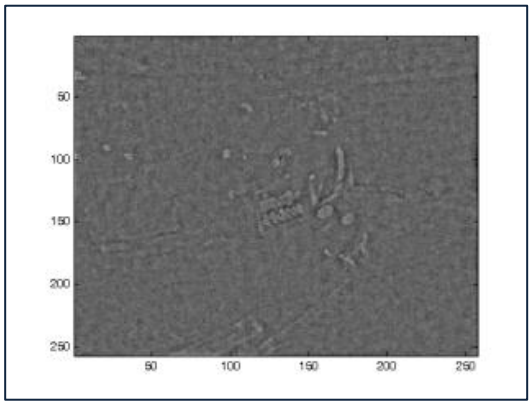
Figure 3.7: (a)



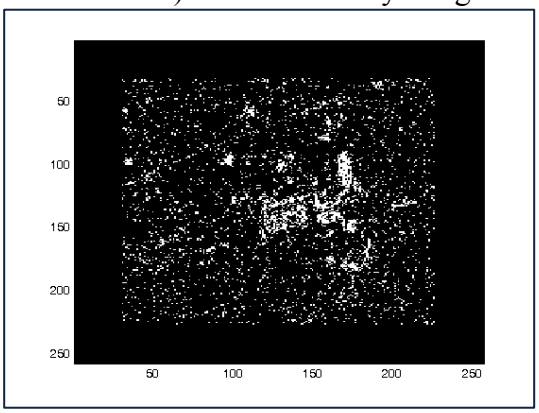
i) Input Image



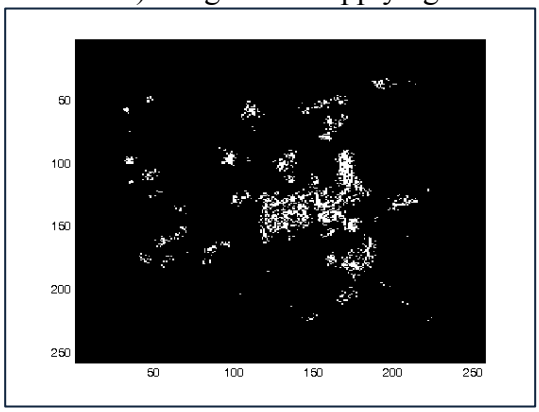
ii) Converted Gray Image

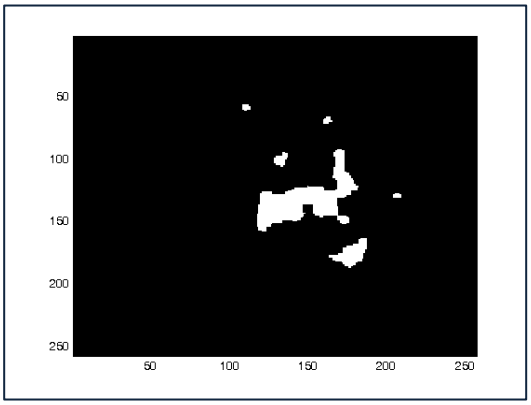


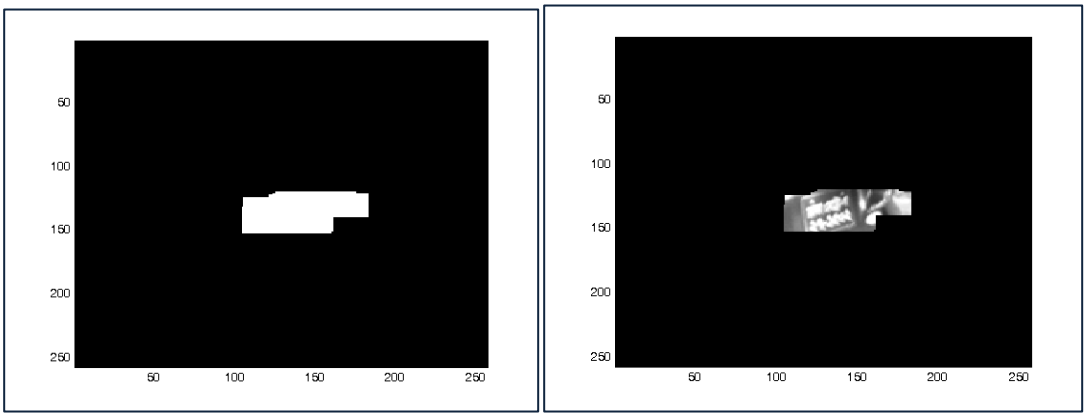
iii) Image After Applying FRIT



iv) Thresholded Image



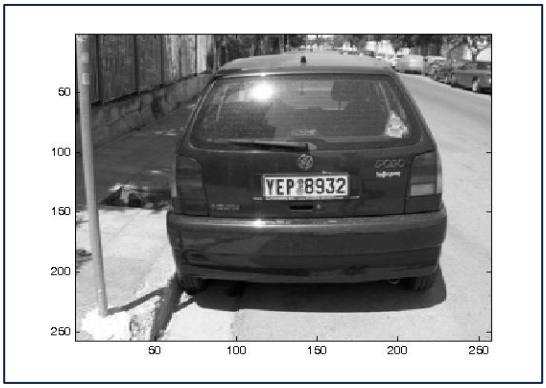




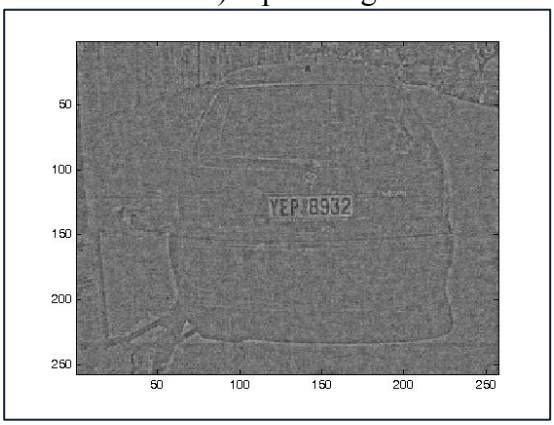
v) Image output after performing morphological operations and verifications.

Figure 3.7: (b)

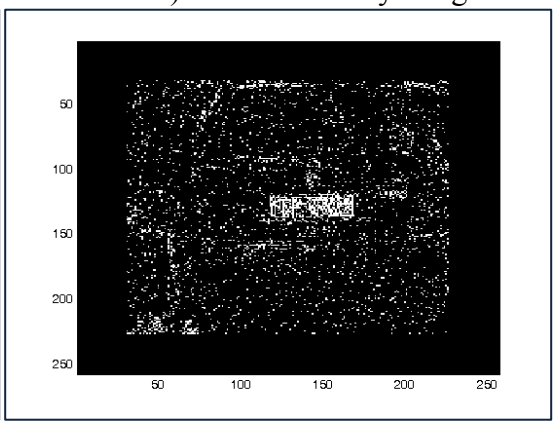
i) Input Image



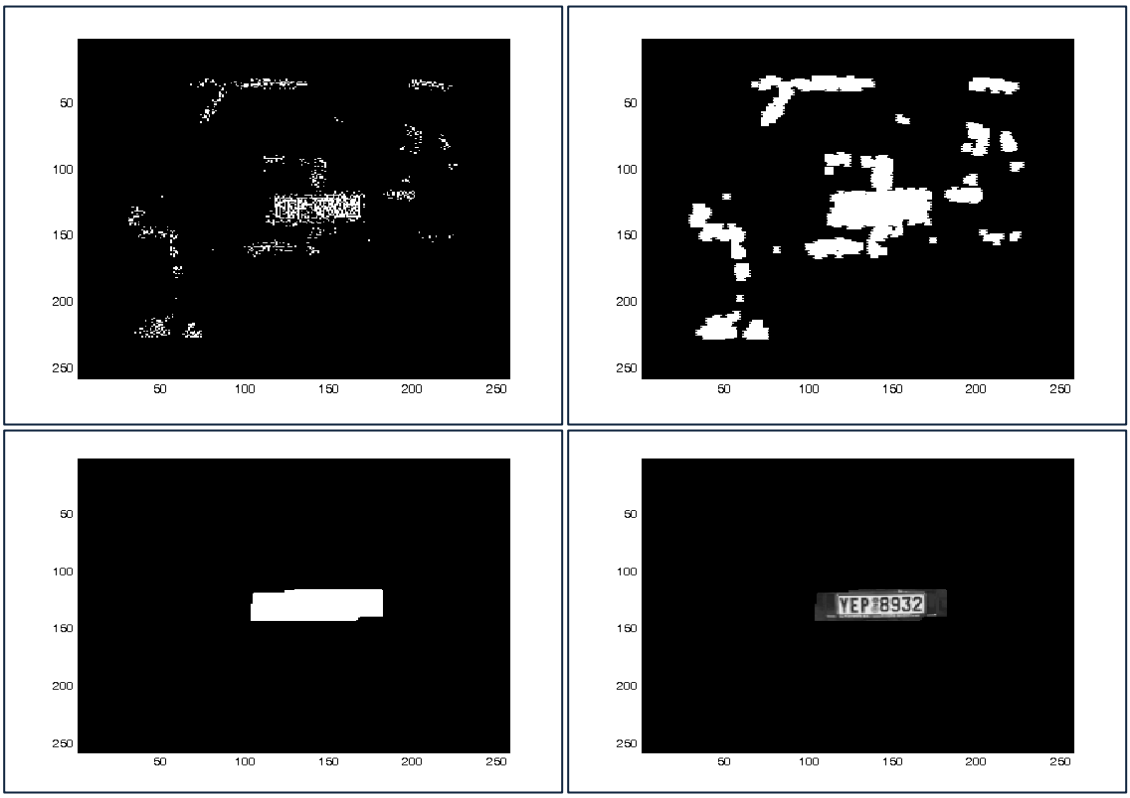
ii) Converted Gray Image



iii) Image After Applying FRIT

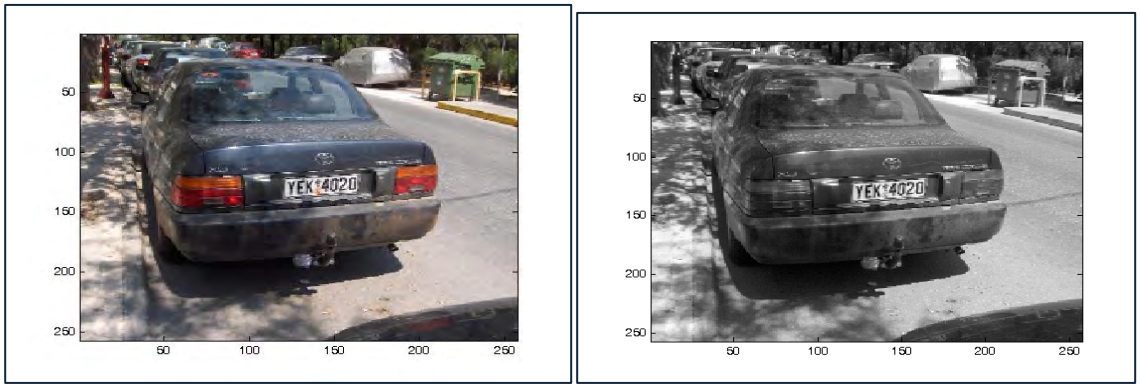


iv) Thresholded Image

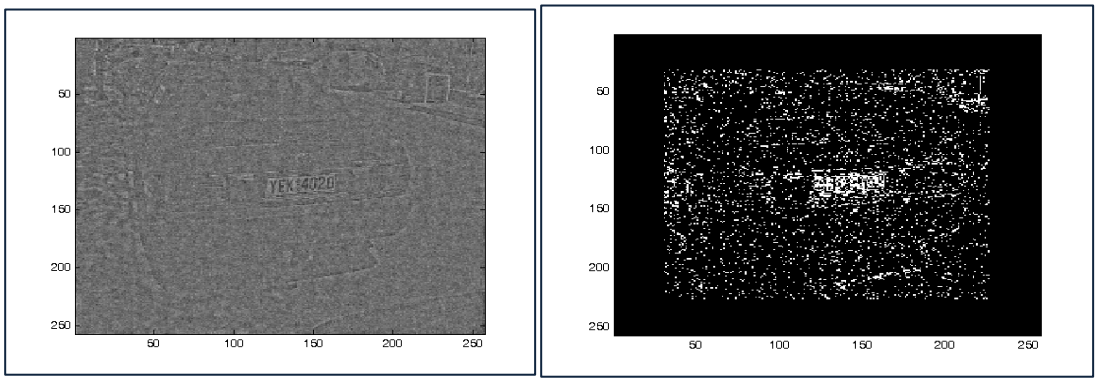


v) Image output after performing morphological operations and verifications.

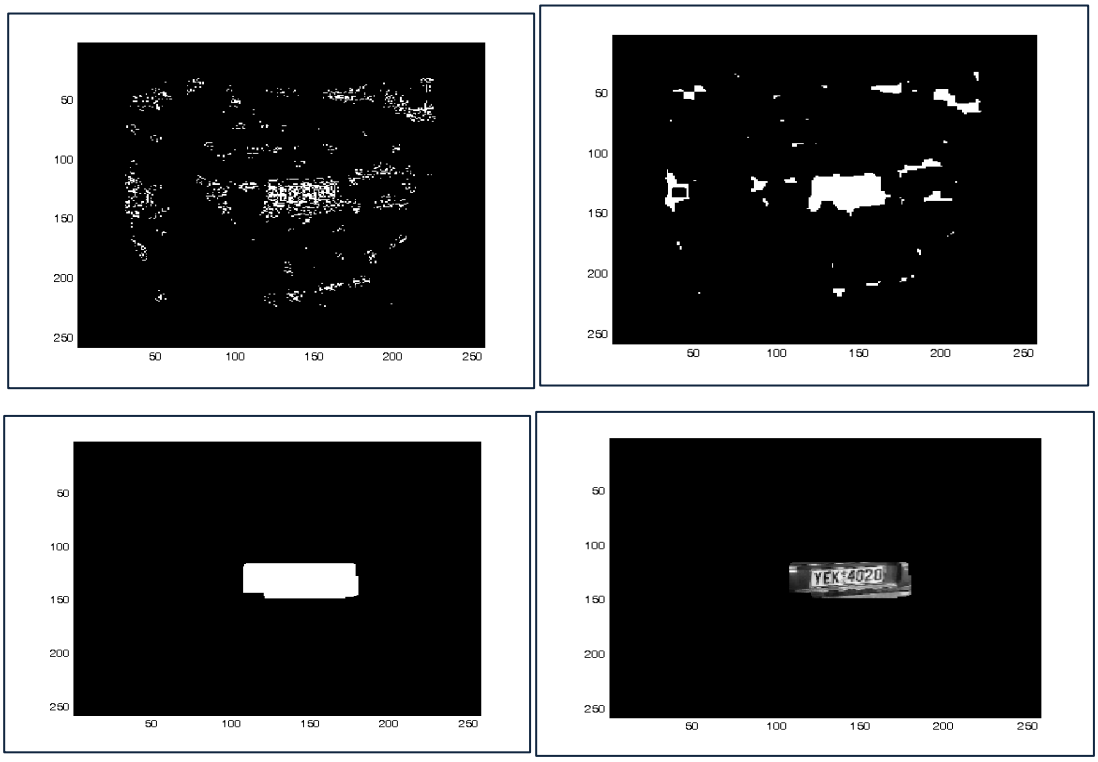
Figure 3.7: (c)



i) Input Image ii) Converted Gray Image

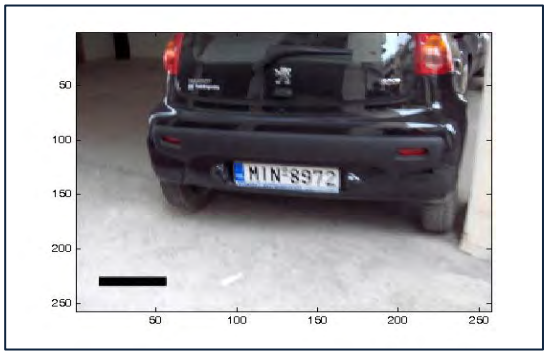


iii) Image After Applying FRIT iv) Thresholded Image

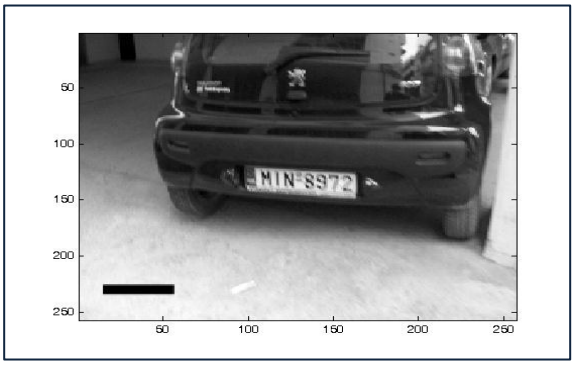


v) Image output after performing morphological operations and verifications.

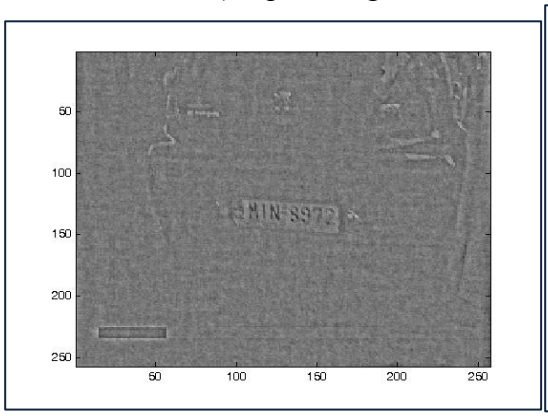
Figure 3.7: (d)



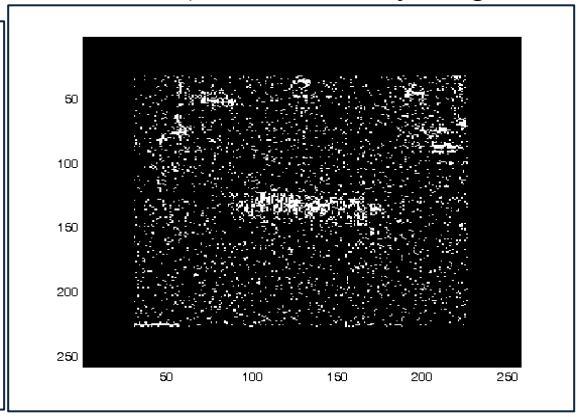
i) Input Image



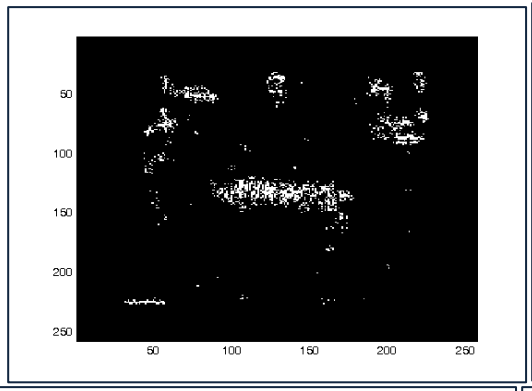
ii) Converted Gray Image

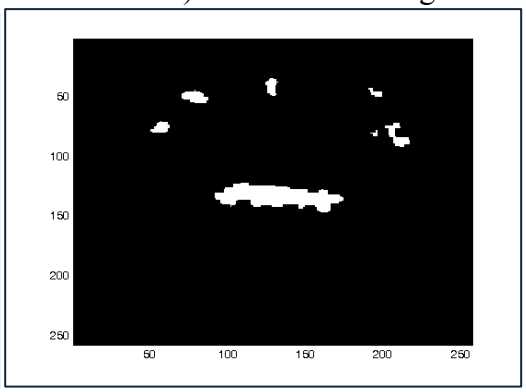


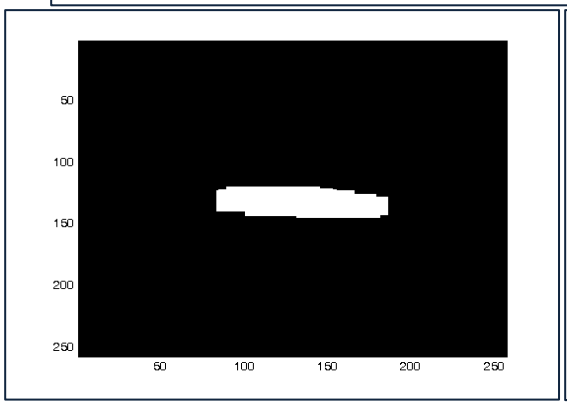
iii) Image After Applying FRIT

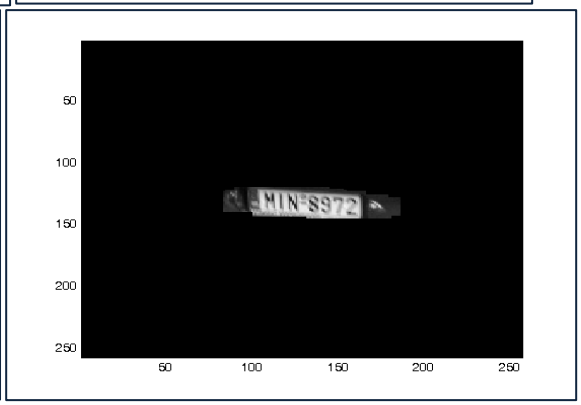


iv) Thresholded Image



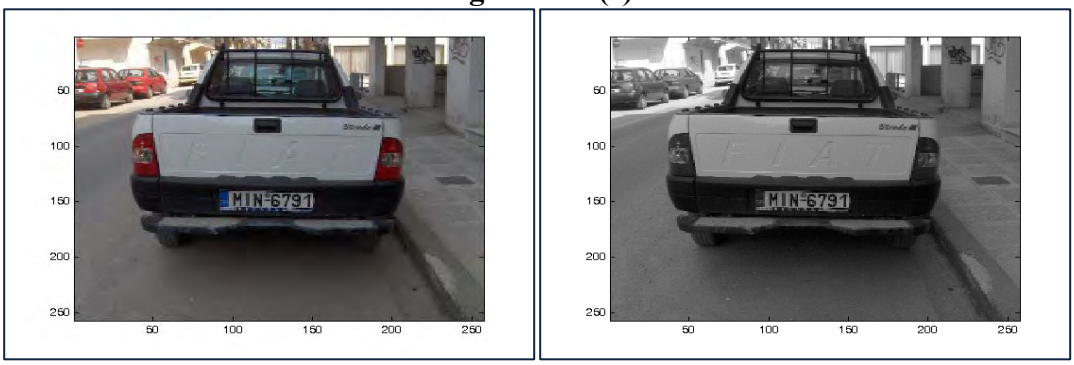




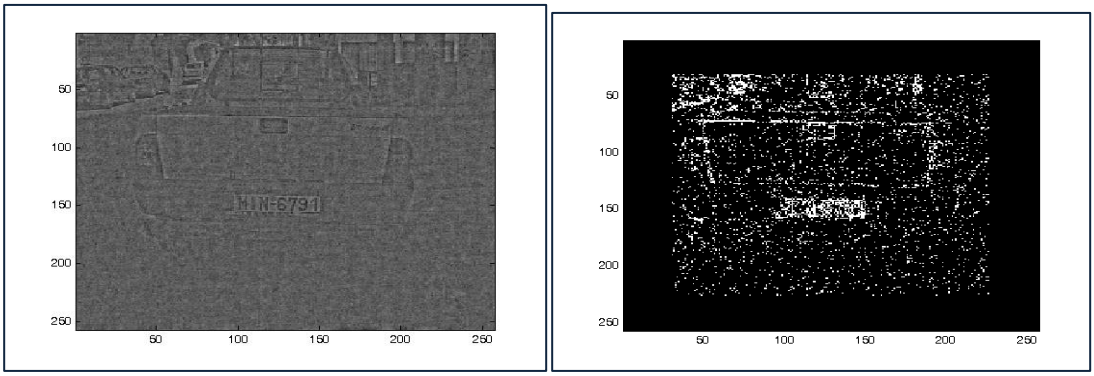


v) Image output after performing morphological operations and verifications.

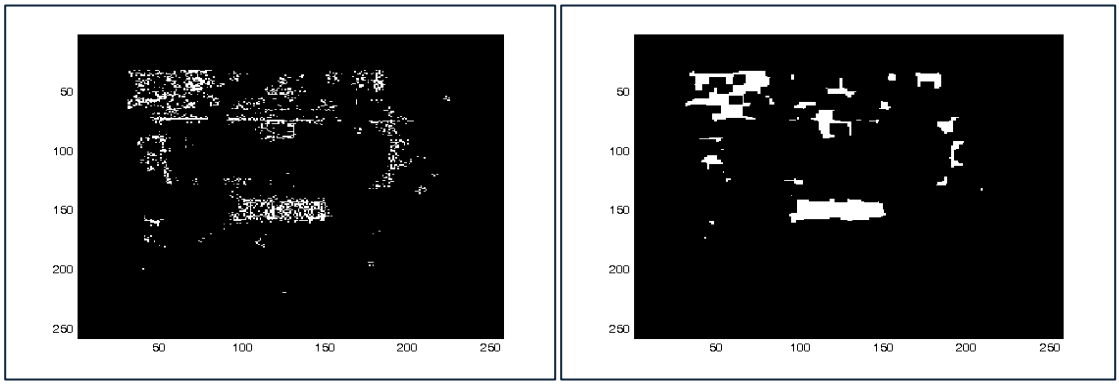
Figure 3.7: (e)

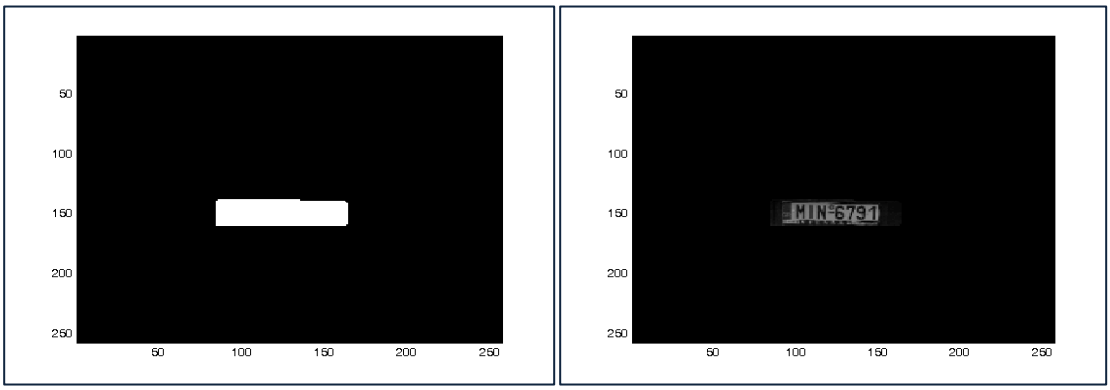


i) Input Image ii) Converted Gray Image



iii) Image After Applying FRIT iv) Thresholded Image





v) Image output after performing morphological operations and verifications.

Figure 3.7: (f)

Figure: 3.7: Several stages of sample images for the LP detection method.

Localizing and detecting a license plate consists these several steps and finding the regions of interest is a key step. After finding RoI, we can verify this/these regions to satisfy that it is a license plate by using methods mentioned above. From our sample images we have found many images with multiples RoI. By filtering them we were able to find the correct region for license plate and also found several regions for several number plates.

3.6 Chapter Summary

In this thesis, the method of a novel technique for Automatic Vehicle License Plate Detection System using Finite Ridgelet Transform is discussed. To detect the license plates by using FRIT we introduced a new thresholding technique for the FRIT coefficients and then perform several operations. We showed all the steps necessary to detect a license plate by several figures and discussed those steps.

Chapter 4

Results and Discussion

4.1 Introduction

This chapter discusses the result, findings and outcome of the thesis. We have discussed all the simulation results and compare them with other methods discussed in the literature.

4.2 Simulation Results and Comparison with Other Systems

The ultimate goal of this thesis is to take an image containing a vehicle's license plate as input and detect the license plate by introducing a new method: Finite Ridgelet Transform (FRIT), to the Automatic Vehicle License Plate Recognition (AVNPR) System.

4.2.1 Overall Performance

We did not impose any restriction on the image as well as the license plate only to see how well our new technique is performed.

We tested our system on a laptop running on Intel Core 2 Duo 1.86GHz Processor and 1GB of RAM by using MATLB R2010a. Total of 140 images from the internet and by capturing devices are taken as sample images. We could detect license plates of 138 sample images which give us a success rate of about**98.60%.** But this could be reached up to 100% since there are few parameters in the process which can increase the performance by alternating values upon some conditions. Moreover if we pre-processed our input image like histogram equalization, brightness enhancement, contrast enhancement, color correction, reducing blurriness, the performance of the system would increase dramatically. The verification techniques, earlier said can be applied to eliminate the unwanted objects detected by the system. The core algorithm along with FRIT to detect license plates only took about **510ms**, which is almost real-time. This

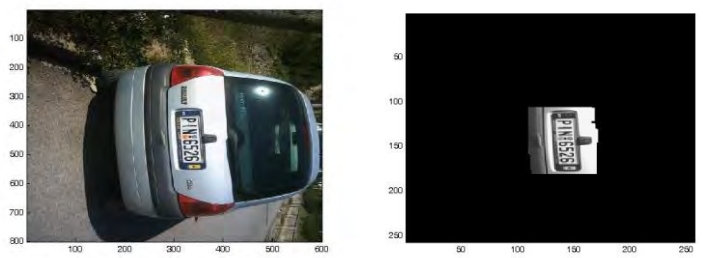
time stamp is for the system which is not optimal. In the process there are many places where we can optimize and make this system more robust. To detect the license plate of all size we first did not impose any restriction on the license plate size, but from the literatures from previously discussed we found that very small sized license plates are useless to AVNPR system since they are poor for character segmentations. Therefore we were only concerned about license plates of size greater than 15x5 pixels. Though this size is also very small we chose it as a benefit of us. Below we represent significant results and outcomes of this newly proposed method and comparison of the proposed method by other existing methods.

4.2.2 MATLAB based simulation results

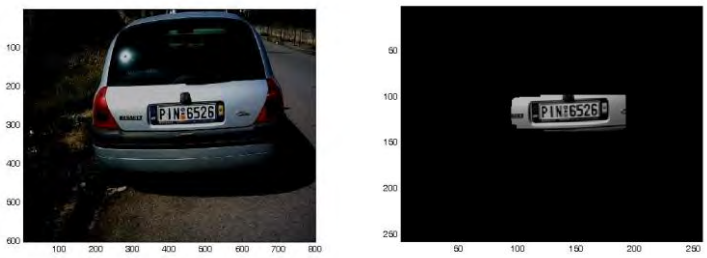
The outcomes of MATLAB simulations are significant. We have simulated the system for a large number of sample input images. The input images were of various kinds like blurred image, image containing picture of dirty and dusty cars, image of multiple cars, low contrast image, etc. The results are discussed below.

Figure 4.1 **a.** shows a sample input image and its output. **b.** shows the same input image but 90 degree rotated and the output is also a license plate but 90 degree rotated, means that the technique is rotation invariant. **c, d, e** are same input images but with lower brightness, lower contrast and higher contrast correspondingly. The output is always the license plate but lower brightness, lower contrast and higher contrast meaning that the technique is much more effective for a poor illumination conditioned input image.

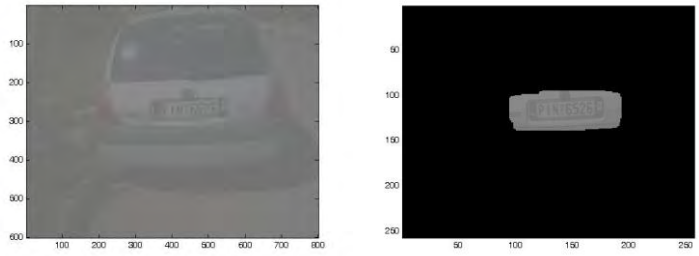
a) Original Image,



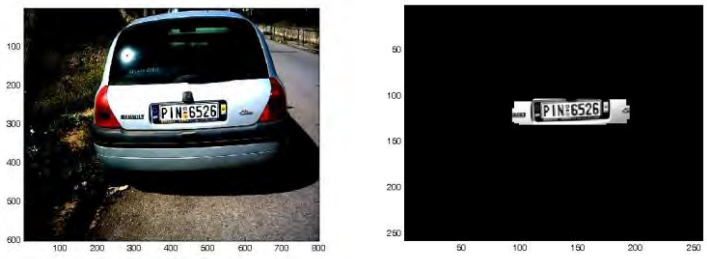
b)Rotated Image



c)Lower Brightness



d) Lower Contrast



e)Higher Contrast

Figure 4.1: License Plate Detection of Same Input Image under Various Conditions

Figure 4.2 a., b**.** below reveals the fact that the technique is effective if we enhance the input image. In both cases at first time license plate was unable to be detected. But after enhancing the input image the license plates were able to detected using the proposed technique.

Figure 4.2:(a-1): Original Image and Output

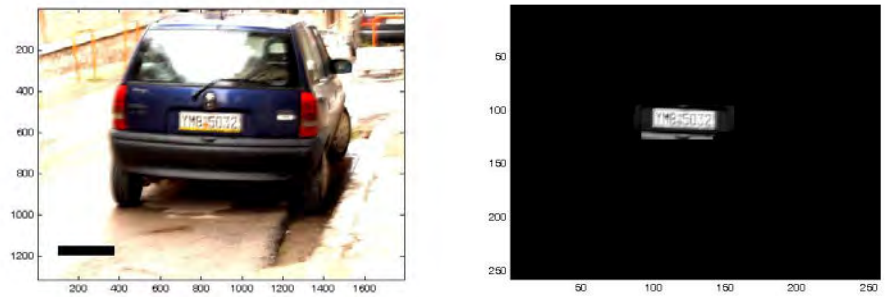


Figure 4.2:(a-2): Enhanced Image and Output

Figure 4.2: (b-1): Original Image and Output

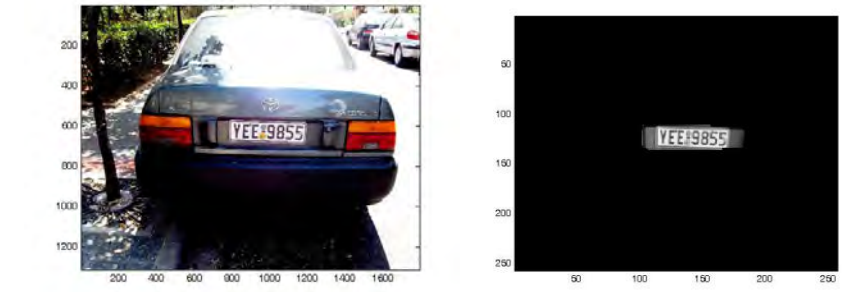
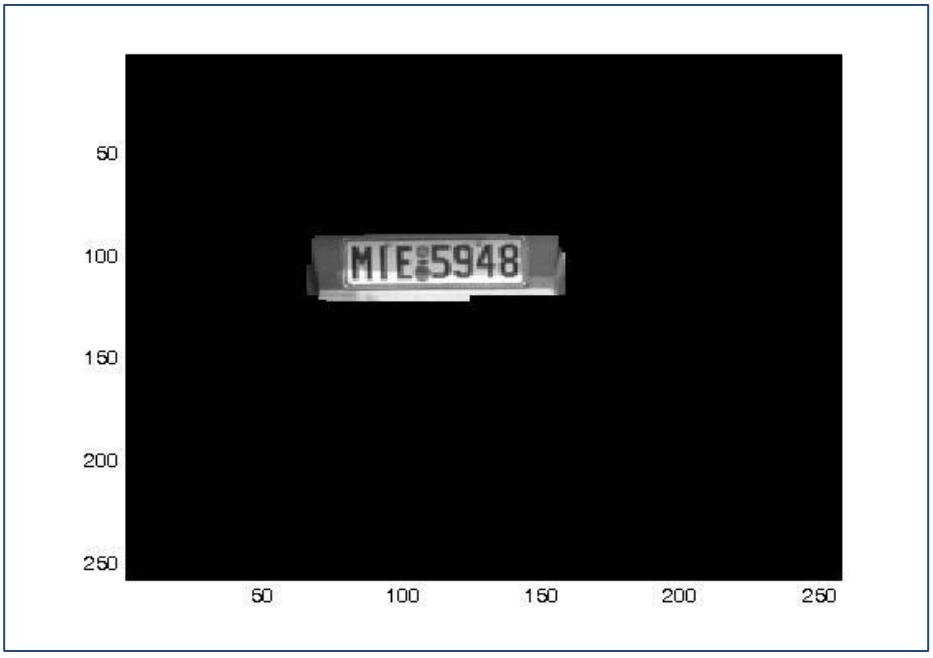


Figure 4.2: (b-2): Enhanced Image and Output

Figure 4.2: License Plate Detection after Enhancing Brightness/Contrast

Figure 4.3 shows an example where the input image contains a license plate shadow on it. It is very difficult for most of the existing methods discussed in the literatures while there is shadow on the license plate. This example shows that our technique is very effective in such cases.

a) Input Image

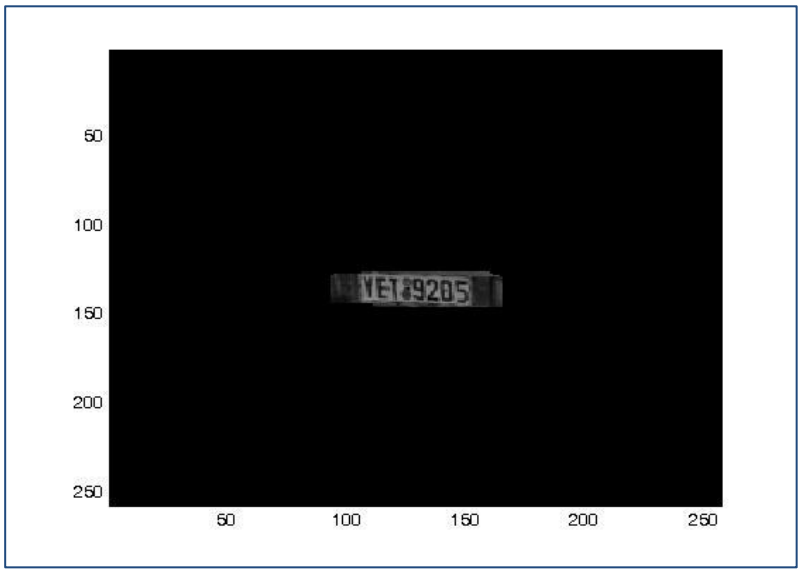


b) Output Image

Figure 4.3: License Plate Detection While Shadows on the Plate

Figure 4.4 shows a detected license plate by using the proposed technique where there is dust and dirt on the license plate. It is almost impossible by the methods discussed in the literatures but as we can see our technique is much more effective here.

a) Input Image

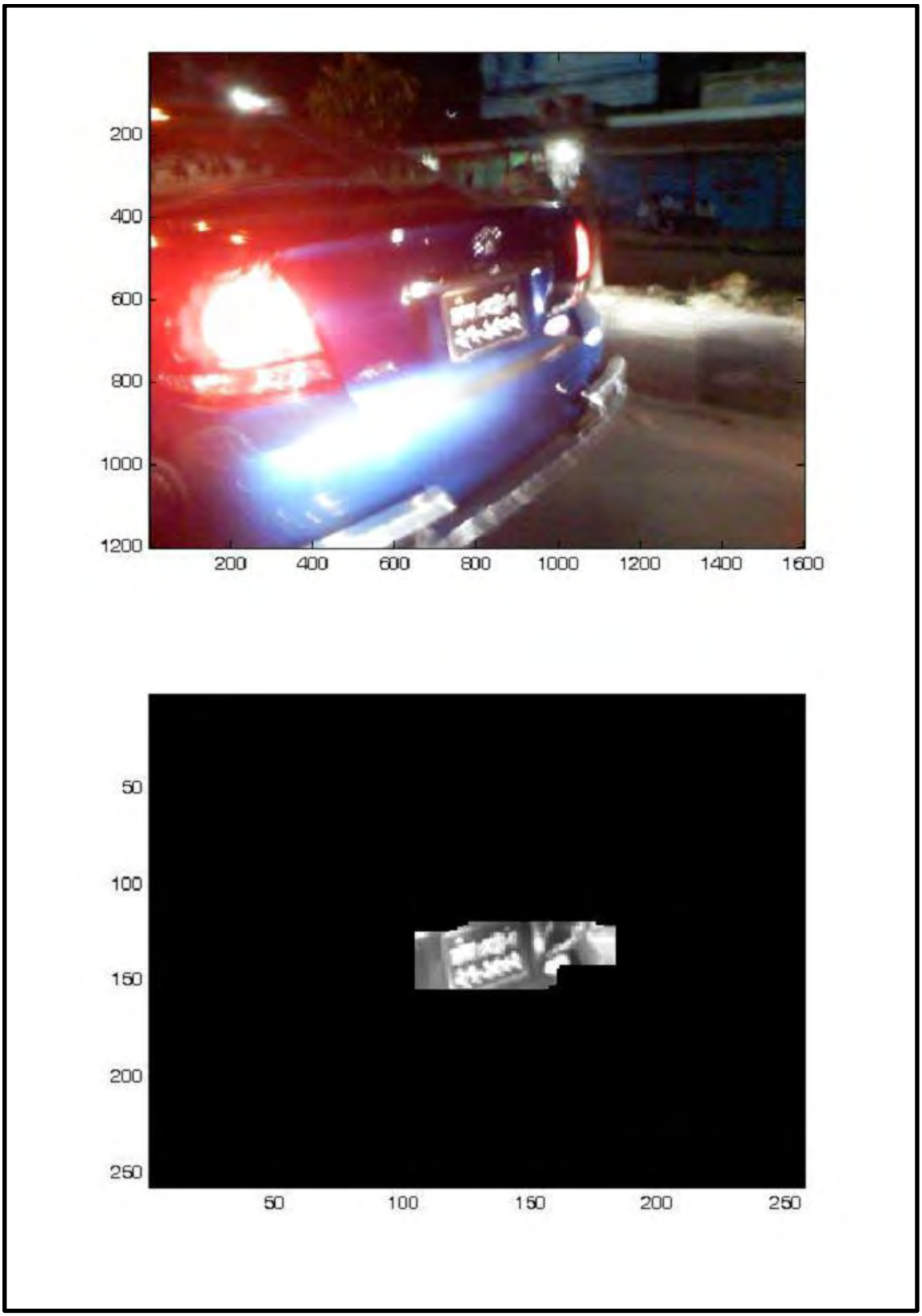


b) Output Image

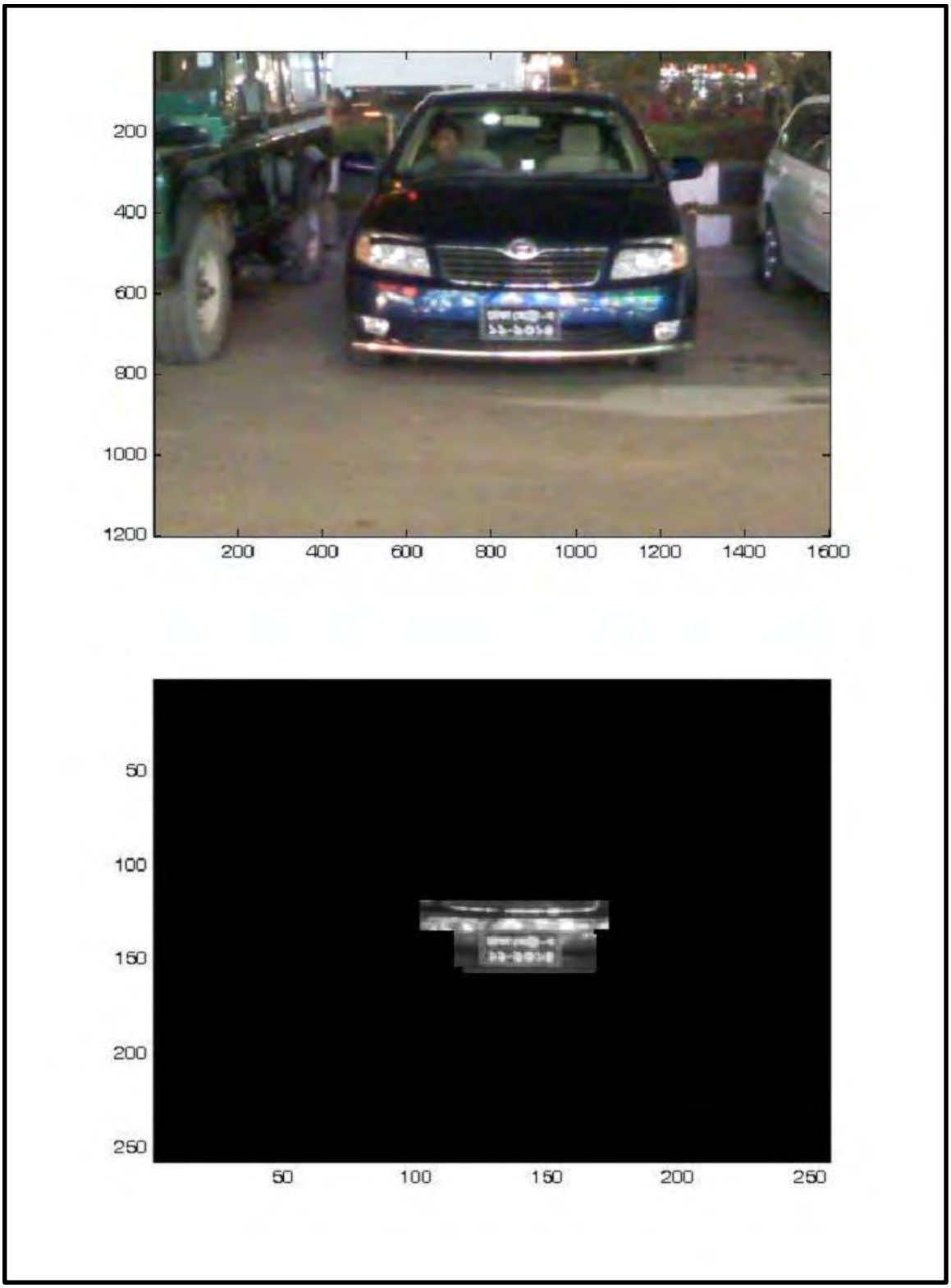
Figure 4.4: Effects on Dirty and Dusty License Plates

Figure 4.5: a.b.c.d.e.f shows examples of detected license plates of some blurred input image. Most of the existing techniques discussed in the literatures suffer from the fact that that in most cases they are unable to detect license plates of blurred input image. Image enhancement is required for detection of license plates from blurry images. This example shows that our technique does not require image enhancement of input images for blurriness, although image enhancement would increase the performance of the technique.

a) Input and Output Image



b) Input and Output Image



c) Input and Output Image

d) Input and Output Image

e) Input and Output Image

f) Input and Output Image

Figure 4.5: Effects on Blurred Image

Figure 4.6shows that our proposed technique is capable of multiple license plate detection from a single input image. Figure 4.6.a shows a sample input image and the detected license plate. Figure 4.6.bhas the same input image but with some extra license plates, and the output shows extra license plates detected with the original one.

Figure 4.6: a)Original Image and the Detected License Plate

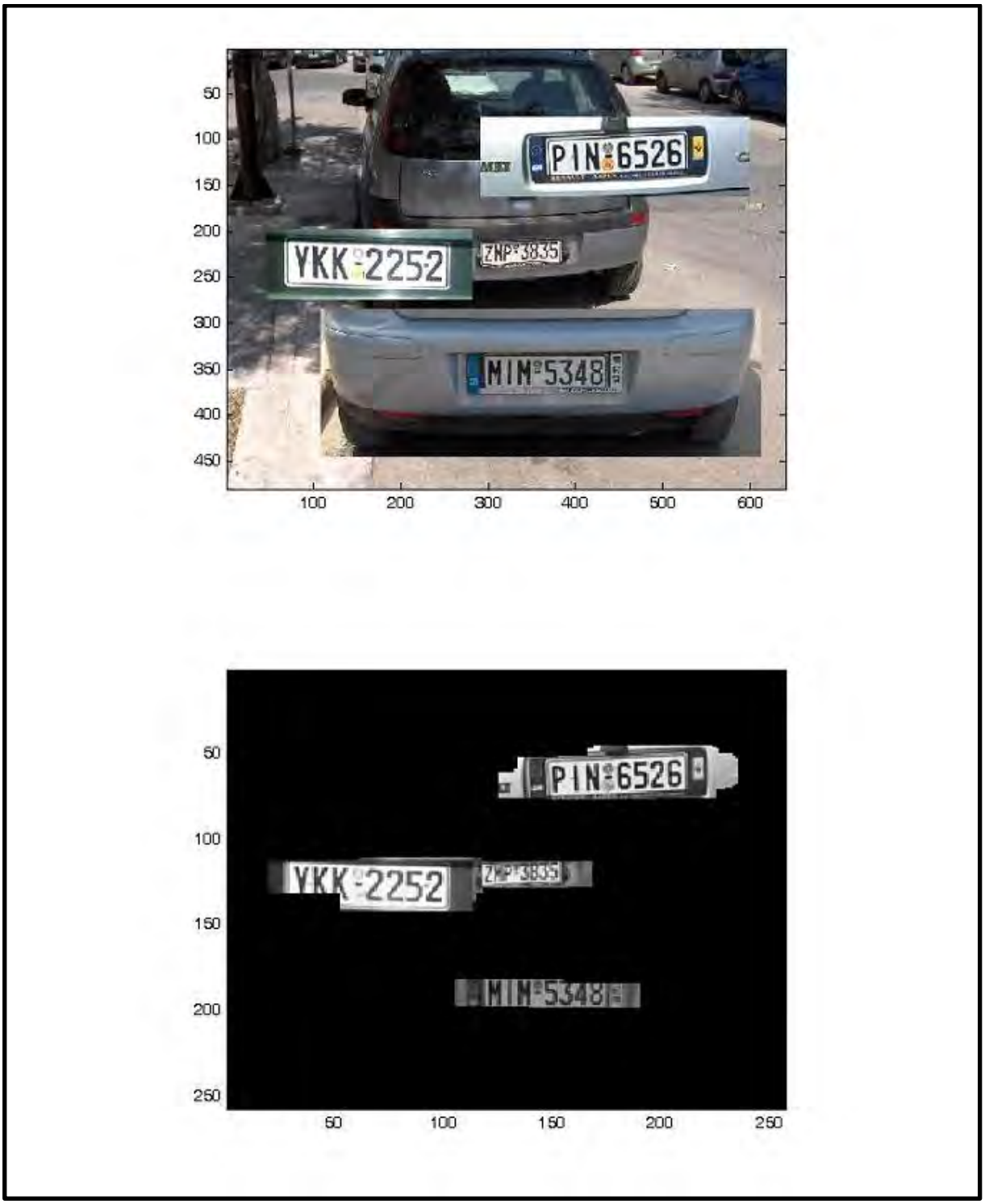


Figure 4.6: b)Image with Multiple License Plates and the Detected Plates

Figure 4.6: Examples of Multiple License Plates Detection

4.2.3 Performance and Time Comparison with Other Systems

Table 4.1 shows license plate detection performance and minimum resolution of various methods in several literatures.

Table 4.1: Plate Detection Performance and Minimum Plate Resolution

References	Size	Success Rate
S. Kim [**10**]	65x20	96.50%
P. Comelli [27]	100x25	84.20%
E. Kayafas [**28**]	61x20	87.80%
Kahraman [**29**]	173x37	98%
K. I. Kim [**30**]	79x38	~90%
N. Zimic [**31**]	100x25	97%
D. Yan [**32**]	100x20	87%
R. S. Venema [**33**]	100x25	75.40%
T. Naito [**34**]	160x80	97%
D. Llorens [**35**]	200x40	88%
Proposed Technique	15x5	98.60

Table 4.1 shows that our method is capable of detecting very much minimum plate size with higher accuracy than that of other researchers.

Table 4.2 shows time required for different methods.

Table 4.2: Elapsed time using different methods for test (In Milliseconds)

Detection Method	Generic Algorithm	Artificial Neural Network	Fuzzy C-means	Support Vector Machine	Method of Feature Salience	Method applying FRIT
Computational Cost	712	1026	878	903	612	510

Table 4.2 shows that computational time for LPD using our method is lower than that of other researchers and it is also in real time.

Table 4.3 shows that summary of the field test results

Table 4.3: Summary of the field test

Case	Working Condition	Result
1	Night	Succeed
2	Day	Succeed
3	Moving Car	Succeed

From the above result as shown in table 4.3 we can see that our proposed method can successfully detect the license plate in various working condition.

4.3 Chapter Summary

In this chapter we have represented the analysis and their results of the proposed techniques in a variety of fashions. We have shown that the proposed technique can detect plates of different sizes, different illumination conditions, rotations, scales, shadows, and the real world noise. These are obviously the plus points among the existing AVNPR systems discussed in the literatures. We had compared our technique with several others of different methods in terms of performance and computational costs, and found that our proposed technique outperforms most of them. Finally we have shown the simulation based results under various conditions and showed how our proposed technique can beat the other existing techniques.

Chapter 5

Conclusion

5. Conclusion

Automatic vehicle number plate recognition (AVNPR) is plays an important role in intelligent transportation system and it has huge number of practical applications such as automatic toll collections, parking fee payment, detection of vehicle crossing speed limits and thereby reducing road accidents etc. LPD is one of the most crucial components of total solution of AVNPR. Literatures have been extensively reviewed and found the challenges in this area due to the varying size, shape and position (rotation) of the plate, varying illumination condition, varying color, type and noises in the image. Again the image may be blurred which is a common scenario for motion compensated pictures. Obstacles such as shadows, dirt and dust may make the system too difficult to detect the license plate. This thesis presents FRIT based effective LPD system that can overcome all of these limitations. Experimental results over a number of benchmark images presented in this thesis shows that it can process the computational work for license plate detection in real time. Detection accuracy of the research work is also compared with that of others researches where it is shown that our work outperforms all the existing techniques. A field test has also been conducted to prove the proper functionality of the proposed technique and test the performance, which shows that our work is also effective in this case.

Suggestion for Future Work:

The author recommends the following suggestions for the enhancement of the research work presented in this thesis.

Character recognition:-Next part of the License plate detection is the character recognition. An efficient algorithm of character recognition can take place to find numbers written in the license plate.

FPGA Implementation: The technique for LPD presented in this thesis can be implemented in FPGA platform for further improvement of the speed and other performance parameters.

References

[**1**] M. Hamid, K. Shohreh, D. Faezeh, D. Fatemeh, **"*An Efficient Features–Based License PlateLocalization Metho",*** 18th International Conference on Pattern Recognition, 2006.

[**2**] A.E. Christos-Nikolaos, A.E. Ioannis and others, **"*License Plate Recognition from Still Images and Video Sequences: A Survey*",** IEEE Transactions on Intelligent Transportation Systems, vol. l.9, no. 3, pp. 377-391 2008.

[**3**] S. A. Boroujeni, **"*Design and implementation of automatic system of image recording of vehicles license plate*"**, Master's thesis, Amirkabir University of Technology(Tehran Polytechnic), 2000.

[**4**] B. Hongliang and L. Changping, **"*A hybrid license plate extraction method based on edge statistics and morphology*"**, 17th International Conference on Pattern Recognition, 2004.

[**5**] M. Sarfraz, M. J. Ahmed, and S. A. Ghazi, **"*Saudi Arabian license plate recognition system",*** International Conference on Geometric Modeling and Graphics, 2003.

[**6**] J. W. Hsieh, S. H. Yu, and Y. S. Chen, **"*Morphology based license plate detection from complex scenes*"**, 16th International Conference On Pattern Recognition, 2002.

[**7**] F. Mart´ın, M. Garc´ıa, and J. L. Alba, **"*New methods for automatic reading of vlps (vehicle license plates",*** Signal Processing Patten Recognition and Application, 2002.

[**8**] V. Kamat and S. Ganesan, **"*An efficient implementation of hough transform for detecting vehicle lcense plate using dsp's",*** 1st IEEE Real-Time Technology and Applications Symposium, 1995.

[**9**] V. Shapiro, D. Dimov, S. Bonchev, V. Velichkov, and G. Gluhchev, ***"Adaptive license plate imageextraction",*** International Conference on Computer Systems and Technologies, 2003.

[**10**] S. Kim, D. Kim, Y. Ryu, and G. Kim, ***"A robust license-plate extraction method under complex image conditions",*** 16th International Conference on Pattern Recognition, 2002.

[**11**] Y. Cui and Q. Huang, ***"Automatic license extraction from moving vehicles",*** International conference on Image Processing, 1997.

[**12**] **W.** Wang, O. Jiang, Zhou, X, W. Wan***, "Car license plate detection based on MSER",*** International Conference on Consumer Electronics, Communications and Networks, pp. 3973 – 3976, 2011.

[**13**] L. Bo, T. Bin,Y. Qingming and W. Kunfeng, ***"A vehicle license plate recognition system based on analysis of maximally stable extremal regions",*** 9th IEEE International Conference on Networking, Sensing and Control, 2012.

[**14**] G. Divya andR. Kumudha,***"License Plate Recognition- A Template Matching Method",*** International Journal of Engineering Research and Applications (IJERA) vol. 3, no. 2, pp.1240-1245, 2013.

[**15**] S. Dheeraj andM. Ajay,***"Vehicle Licence Plate Recognition Using Gaussian Hermite Moments and Wavelets"***, International Journal of Advanced Research in Computer Science and Software Engineering. Vol. 3, no 7, 2013.

[**16**] C. Dong, G. Dongbing, G. Hua and S. Junxi, ***"License plate detection algorithm based on gentle AdaBoost algorithm with a cascade structure",*** IEEE International Conference on Robotics and Biometrics, 2009.

[**17**] L. Liang, H. Youngjoon and H. Hernsoo, ***"License plate detection method using vertical boundary pairs and geometric relationships",*** Computer Engineering and Technology (ICCET), 2010.

[**18**] W. Ying, L. Yue, Y. Jingqi, Z. Zhenyu, D. von andS. Pengfei, ***"An Algorithm for License Plate Recognition Applied to Intelligent Transportation System",*** IEEE Transactions on Intelligent Transportation Systems, vol.12 , no. 3, p.830-845, 2011.

[**19**] Stephen Johnson, "***Stephen Johnson on Digital Photography",*** 16th International Conference on Pattern Recognition, *1999.*

[20] Dwayne Phillips, "***Image Processing in C Analyzing and Enhancing Digital Image*****"**, 14th International Conference on Pattern Recognition, 1997.

[21] Candes E J, Donoho D L***, "Ridgelets: a key to higher-dimensional intermittency",*** Philos Trans R Soc A-Math PhysEngSci, 1999.

[22] Starck J -L, Candes E J, Donoho D L. "***The curvelet transform for image de-noising",*** IEEE Trans on Image Process, 2002.

[23] Minh, N. Do and Vetterli, M., **"*The Finite Ridgelet Transform for Image Representation"***, IEEE Transactions on image processing, vol.12, no.1, 2003.

[24] Matus F, Flusser J. "***Image representation via a finite Radon transform",*** IEEE Trans Pattern Anal Mach Intell, 1993.

[25] Liu Y.X, Peng Y.H., QuHuaiJing& Yin Yong. "***Energy-based adaptive orthogonal FRIT and itsapplication in image denoising",*** China, 2012.

[26] Xiao X K, Li S F. "***Edge-preserving image denoising method using Curvelettransform",*** J Commun(in Chinese), pp:9—15, 2004.

[27] P. Comelli, P. Ferragina,M. N. Granieri, and F. Stabile, ***"Optical recognition of motor vehiclelicense plates,"*** IEEE Trans. Veh. Technol., vol. 44, no. 4, pp. 790–799, Nov. 1995.

[28] C. Anagnostopoulos, I. Anagnostopoulos, E. Kayafas, and V. Loumos, ***"A license plate recognition system for intelligent transportation system***

applications," IEEE Trans. Intel. Transp. Syst., vol. 7, no. 3, pp. 377–392, Sep. 2006.

[**29**] Kahraman, F., Kurt, B., Gökmen, M., "***License Plate Character Segmentation Based on the Gabor Transform and Vector Quantization",*** vol. 2869, New York: Springer-Verlag, pp. 381–388, 2003.

[**30**] K. I. Kim, K. Jung, and J. H. Kim, "***Color Texture-Based Object Detection:An Application to LicensePlate Localization"***, New York: Springer-Verlag, vol. 2388, pp. 293–309, 2002.

[**31**] N. Zimic, J. Ficzko, M. Mraz, and J. Virant, ***"The fuzzy logic approach to the car number plate locating problem,"*** in Proc. IIS, Grand Bahama Island, The Bahamas, , pp. 227–230, 1997.

[**32**] D. Yan, M. Hongqing, L. Jilin, and L. Langang, ***"A high performance license plate recognition system based on the web technique,"*** in Proc. Conf. Intell. Transp. Syst, pp. 325–329, 2001.

[**33**] J. A. G. Nijhuis, M. H. terBrugge, K. A. Helmholt, J. P. W. Pluim, L. Spaanenburg, R. S. Venema,and M. A.Westenberg, ***"Car license plate recognition with neural networks and fuzzy logic,"*** in Proc. IEEE Int. Conf. Neural Netw, vol. 5, pp. 2232–2236, 1995.

[**34**] T. Naito, T. Tsukada, K. Yamada, K. Kozuka, and S. Yamamoto, ***"Robust license-plate recognition method for passing vehicles under outside environment,"*** IEEE Trans. Veh.Technol., vol. 49, no.6, pp. 2309–2319, Nov. 2000.

[**35**] D. Llorens, A. Marzal, V. Palazon, and J. M. Vilar, "***Car License Plates Extraction and Recognition Based on Connected Components Analysis and HMM Decoding"***, New York: Springer-Verlag, vol. 3522, pp. 571–578, 2005.

APPENDIX

MATLAB Codes:

/**/

avnpr_final_a_v4.m

```
%% FILE List
% avnpr_final_a_v4.m
% frito_avnpr.m
% process_ri_row.col.m
% ifrito_avnpr.m
% calculate_histo.m
% find_cutoff_point.m
% erode_imrecs.m
% mask_erode_imrecs3.m
% mask_erode_imrecs.m
% find_rectangle.m
% process_main_image2.m
% process_main_image3.m
% frat_frito_avnpr.m
% isprime.c
% mean_frito_avnpr.m
% isdyadic_frito_avnpr.m
% bestdir_frat_frito_avnpr.m
% angle_bestdir_frat_frito_avnpr.m
% fratc.c
% ifratc.c
% dwtmode_frito_avnpr.m
% wavedecc_frito_avnpr.m
% waverecc_ifrito_avnpr.m
% ifrat_ifrito_avnpr.m
%% Image acquisition
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%->>>>>100,7,9,11,15,1,2,12,10
% myImg3 = imread('Images/100.JPG');%****
myImg3 = imread('Images/100 (1).JPG');%****
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%->>>>>200,1,2,18,13,15,9,5
% myImg3 = imread('Images/200.jpg');%****
% myImg3 = imread('Images/200 (1).jpg');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%%%->>>>>4,4.2,4.3,4.4,4.5,
5,8,8.2,12,12.2,12.3,12.4,12.5,15,16,17,23*,26,26.2,27,28,30,31,32,34,3
5,37,38,42,45,46,49,50,51,53,54,55*,
```

```
%%%%-
>>>>>57,58,59,60,61,62,64,65,68,70,71,73,74,79,80,81,82,84,48.2*,75.2*,
6,10,11,76,78,600, 21.2, 21.3,101.2*,21*,48*,75*,85,101*,
% myImg3 = imread('Images/600.jpg');
% myImg3 = imread('Images/600 (78).JPG');
% myImg3 = imread('Images/600 (100).BMP');
% myImg3 = imread('Images/600 (101).BMP');
% myImg3 = imread('Images/600 (101.2).BMP');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%%%->>>>>1,2,3,5*,5.2,11*,11.2,13
% myImg3 = imread('Images/700 (13).jpg');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
figure(1),
imagesc(myImg3);%,colormap('gray');
title('Input Image');
%
%
%% Image Processing
[p,q,r] = size(myImg3);
if((p~=257) || (q~=257))
%     display('Image size is not 257x257')
    p=257;
    q=257;
    myImg4 = imresize(myImg3, [p q]);
%     display('Image size is changes to 257x257')
else
    myImg4 = myImg3;
%     display('Image size is 257x257')
end;


if (ndims(myImg4) == 3) % 'ndims' built in function
    R_myImg4 = myImg4(:,:,1);
    G_myImg4 = myImg4(:,:,2);
    B_myImg4 = myImg4(:,:,3);
    myImg2 = rgb2gray(myImg4); % 0.2989 * R + 0.5870 * G + 0.1140 * B
    myImg1 = double(myImg2);
    im3 = myImg1/255;
elseif (ndims(myImg4) == 2)
       myImg2 = myImg4;
       myImg1 = double(myImg4);
       im3 = myImg1/255;
end;
for j_row=1:1:257
    for i_col=1:1:257
        if (im3(j_row,i_col)>=1)
            im3(j_row,i_col)=1;
        end;
        if (im3(j_row,i_col)<=0)
           im3(j_row,i_col)=0;
        end;
```

```
    end;
end;
%
im = im3;
figure(2),%************************************************************
**************************************************
imagesc(im, [0,1]), colormap('gray'),
title('Converted Gray Image');
%% Wavelet parameters FRITO
wname = 'haar';
maxR = 258;
DWTMODE = 'zpd';
%             'zpd' ,            'Zero Padding'
%             {'sym','symh'} ,   'Symmetrization (half-point)'
%             'symw' ,           'Symmetrization (whole-point)'
%             {'asym','asymh'} , 'Antisymmetrization (half-point)'
%             'asymw' ,          'Antisymmetrization (whole-point)'
%             'sp0' ,            'Smooth Padding of order 0'
%             {'spd','sp1'} ,    'Smooth Padding of order 1'
%             'ppd' ,            'Periodized Padding'
%             'per' ,            'Periodization'
%
%
%% FRITO >>>>> FRITO >>>>> FRITO >>>>> FRITO >>>>> FRITO >>>>> FRITO
>>>>> FRITO >>>>> FRITO >>>>> FRITO >>>>> FRITO >>>>> FRITO >>>>>
%***********************************************1111111111111111111
%*********************************************
[ri_fritm, l_fritm, m_fritm] = frito_avnpr(im,wname,DWTMODE);
[ri_row,ri_col] = size(ri_fritm);
process_ri_row_col_flag = 1;
tr = 0.0500;
tc = 0.0500;
G=1;
r_m=1;
c_m=1;
r_start = 1;
r_end = ri_row;
c_start = 1;
c_end = ri_col-1;
[ri_fritm,~,~,~,~,~,~] =
process_ri_row_col(ri_fritm,ri_row,ri_col,process_ri_row_col_flag,tr,tc
,G,r_m,c_m,r_start,r_end,c_start,c_end);
%
%% IFRITO >>>>> IFRITO >>>>> IFRITO >>>>> IFRITO >>>>> IFRITO >>>>>
IFRITO >>>>> IFRITO >>>>> IFRITO >>>>> IFRITO >>>>> IFRITO >>>>> IFRITO
>>>>>
if(m_fritm >= 100)
    m_fritm2 = 150;
else
    m_fritm2=m_fritm;
end;
imrec_frito2 = ifrito_avnpr(ri_fritm, l_fritm, m_fritm, wname,DWTMODE);
```

```
%% IMRECFRITO >>>>> IMRECFRITO >>>>> IMRECFRITO >>>>> IMRECFRITO >>>>>
IMRECFRITO >>>>> IMRECFRITO >>>>> IMRECFRITO >>>>> IMRECFRITO >>>>>
IMRECFRITO
%*******************************************************4444444444444444
44444444444444444444444444444
% making [0,255]
im3 = im3.*255;
imrec_frito2 = imrec_frito2.*255;
% Rounding and making integer - [0,255]
im3 = round(im3);
imrec_frito2 = round(imrec_frito2);
% Limitting - [0,255]
for j_row=1:1:257
    for i_col=1:1:257
        if(im3(j_row,i_col)>=255)
            im3(j_row,i_col) = 255;
        end;
        if(im3(j_row,i_col)<=0)
            im3(j_row,i_col) = 0;
        end;
    end;
end;
for j_row=1:1:257
    for i_col=1:1:257
        if(imrec_frito2(j_row,i_col)>=255)
            imrec_frito2(j_row,i_col) = 255;
        end;
        if(imrec_frito2(j_row,i_col)<=0)
            imrec_frito2(j_row,i_col) = 0;
        end;
    end;
end;

figure(3),%***************************************************************
************************************************************
imagesc(imrec_frito2, [0,255]), colormap('gray'),
title('Image After Applying FRIT');

% Thresholding -
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
set_value1 = 255;
set_value2 = 128;
set_value3 = 0;
FORGET_IT1 = -50;
image_row = 257;
image_col = 257;
%find cutoff point-
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
hist_imrecfrito2=calculate_histo(imrec_frito2);
percent_cutoff_point1 = 0.0500;
percent_cutoff_point2 = 0.9350;
% percent_cutoff_point1 = 0.01199;
% percent_cutoff_point2 = 0.8700;
```

```
cutoff_point1 =
find_cutoff_point(imrec_frito2,percent_cutoff_point1,image_row,image_co
l);
cutoff_point2 =
find_cutoff_point(imrec_frito2,percent_cutoff_point2,image_row,image_co
l);
for j_row=1:1:257
    for i_col=1:1:257
        if ((imrec_frito2(j_row,i_col) >= cutoff_point1) &&
(imrec_frito2(j_row,i_col) <= cutoff_point2))
            imrec_frito2(j_row,i_col) = set_value3;
        else
            imrec_frito2(j_row,i_col) = set_value1;
        end;
    end;
end;

for j_row=1:1:257
    for i_col=1:1:257
        if((j_row <= 30) || (j_row >= 227))
%         if((j_row <= 15) || (j_row >= 242))
            imrec_frito2(j_row,i_col) = 0;
        end;
        if((i_col <= 30) || (i_col >= 227))
%         if((i_col <= 15) || (i_col >= 242))
            imrec_frito2(j_row,i_col) = 0;
        end;
    end;
end;
% H = fspecial('unsharp',0.5);
% H = fspecial('gaussian',[3 3],0.3);
% sharpened = imfilter(B,H,'replicate');
% % B=medfilt2(sharpened,[3 3]);
% SE = strel('rectangle',[40 30]);
% BW2 = imerode(imrec_frito98,SE);
% BW3 = imdilate(BW2,SE);

figure(4),%************************************************************
***********************************************************
imagesc(imrec_frito2, [0,255]), colormap('gray'),
title('Thresholded Image');

%Thresholded Eroded Image -
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
erode_value = set_value1; % 255
SE = 8;
            % SE:8-> 9x9->max_erode_threshold = 80
            %       0   0   0 -> 1st check row wise
            %       0   1   0 -> 2nd check row wise
            %       0   0   0 -> 3rd check row wise
erode_threshold = 70;
imrec_frito3 =
erode_imrecs(imrec_frito2,erode_value,erode_threshold,SE,image_row,imag
e_col);% imrecFRITO3: ErodedImage
```

```
figure(5),%***********************************************************
********************************************************
imagesc(imrec_frito3, [0,255]), colormap('gray'),
title('Performing Morphological Operation-1');


% Masked Dilated Image-
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
erode_value = set_value3;
SE = 18;%5x5
imrec_frito3_2 =
mask_erode_imrecs3(imrec_frito3,erode_value,SE,image_row,image_col);%
imrecFRITO3:


% Masked Eroded Image Horizontal-
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
erode_value = set_value1;
SE = 6;%mask = [1 1 1 1 1 1 1];
imrec_frito9_4_1 =
mask_erode_imrecs(imrec_frito3_2,erode_value,SE,image_row,image_col);%
imrecFRITO3:


% Masked Eroded Image Vertical-
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
erode_value = set_value1;
SE =7;%mask = [1;1;1;1;1;1;1];
imrec_frito9_4_2 =
mask_erode_imrecs(imrec_frito9_4_1,erode_value,SE,image_row,image_col);
% imrecFRITO3:



% Thresholded Erosion-
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
erode_value = set_value1; % 255
SE = 8;
erode_threshold = 50;
imrec_frito9_5 =
erode_imrecs(imrec_frito9_4_2,erode_value,erode_threshold,SE,image_row,
image_col);% imrecFRITO3: ErodedImage
figure(6),%***********************************************************
********************************************************
imagesc(imrec_frito9_5, [0,255]), colormap('gray'),
title('Performing Morphological Operation-2');
```

```
% Find Minimum Width Rectangle-
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
check_value = set_value1;
write_value = set_value1;
width = 21;
height = 7;
imrec_frito9_5_2 =
find_rectangle(imrec_frito9_5,image_row,image_col,check_value,write_val
ue,width,height);



% [cc_imrec_frito9_5,num_ccimrec_frito9_5]=bwlabel(imrec_frito9_5,8);


% Masked Dilation Horizontal-
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
erode_value = 0;
SE = 12;% 1x31
imrec_frito9_6 =
mask_erode_imrecs3(imrec_frito9_5_2,erode_value,SE,image_row,image_col)
;% imrecFRITO3:



% Masked Dilation Vertical-
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
erode_value = 0;
% SE =20;%mask = [1;1;1;1;1];
% SE = 26;%mask = [1;1;1;1;1;1];
% SE = 7;%mask = [1;1;1;1;1;1;1];
SE = 22;%mask = [1;1;1;1;1;1;1;1;1]
% SE = 32;%11x1
imrec_frito9_7 =
mask_erode_imrecs3(imrec_frito9_6,erode_value,SE,image_row,image_col);%
imrecFRITO3:
figure(7),%************************************************************
*********************************************************
imagesc(imrec_frito9_7, [0,255]), colormap('gray'),
title('Performing Morphological Operation-3');


% Merge into main image-
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
imrec_frito20 = imrec_frito9_7;
% nchk_value = set_value1;
chk_value = set_value3;
wrt_value = FORGET_IT1;
enc_flag = 0;
enc=10;
```

```
th_flag = 0;
th_val3=65;
val3=0;
th_val4=66;
val4=255;
nth_flag=0;
nth_val5=45;
val5=0;
nth_val6=46;
val6=255;
img1 =
process_main_image2(imrec_frito20,im3,chk_value,wrt_value,image_row,ima
ge_col,enc_flag,enc,th_flag,th_val3,val3,th_val4,val4,nth_flag,nth_val5
,val5,nth_val6,val6);
% img1=img1.*(3/2);
figure(8),%**************************************************************
**************************************************
imagesc(img1, [0,255]), colormap('gray'),
title('Output Image');
```

/***/

frito_avnpr.m

```
function [r, l, m] = frito_avnpr(a, wname,DWTMODE)
% tic
% mlock;
% A_AA = 100000000;
% FRITO    Orthonormal finite ridgelet transform
%   [r, l, m] = frito(a, wname)
%
% Input:
%   a:  image matrix of size P by P, P is a prime number
%   wname:  wavelet name
%
% Output:
%       r:      ridgelet coefficients in a (P-1) by (P+1) matrix,
%       one column for each direction
%   l:  structure of the wavelet decomposition that is
%       needed for reconstruction
%   m:  normalized mean value of the image
%
% Note:
%   This implement the orthonormal version of finite ridgelet
%   transform.  The result (P-1)x(P+1) coefficients in r together
%   with m makes up total of exactly PxP coefficients.
%
%   However there is a restriction on the size P: P+1 must be dyadic
%   number (the method can easily be extended for P = 2^n - m, though).
%   The typical size of the input image for FRITO is 257 by 257.
%
%       The wavelet transform takes the maximum posible number of
```

```
%   decomposition levels.
%
% See also: IFRITO, FRIT
% load im2;
% a = im2;
% wname ='db4';
if ndims(a) ~= 2 % 'ndims' built in function
    error('Input must be a matrix of 2 dimensions');
end

[p, q] = size(a); % 'size' is a built-in function.
if (p ~= q) | ~isprime(p)
    error('Input must be a P by P matrix, P is a prime number')
end

% Subtract the DC component
m = mean_frito_avnpr(a(:));
a = a - m;
% Normalize for unit norm
m = p * m;

% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST%
TEST% TEST% TEST% TEST% TEST% TEST
% a = a.*(3/2);
% for j_row=1:1:257
%     for i_col=1:1:257
%         if (a(j_row,i_col)>=1)
%             a(j_row,i_col)=1;
%         end;
%     end;
% end;
% end of test% end of test% end of test% end of test% end of test% end
of test% end of test% end of test

% Finite Radon transform
ra = frat_frito_avnpr(a);
% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST%
TEST% TEST% TEST% TEST% TEST% TEST
% ri_row = 256;
% ri_col = 258;
% [ra,~,~,~,~,~,~] = process_ri_row_col(ra,ri_row,ri_col);
% [ra,~,~,~] = process_ri_col(ra,ri_row,ri_col);
% ra = ra.*2;
% ra = ra.*(5/4);
% cutoff_point_ra_frito_1 = find_cutoff_point_ri(ra,0.9999,256,258);%
this is greater than the below
% cutoff_point_ra_frito_2 = find_cutoff_point(ra,0.9999,256,258);
% cutoff_point_ra_mean =
(cutoff_point_ra_frito_1+cutoff_point_ra_frito_2)/2;
% for j_row=1:1:257
%     for i_col=1:1:258
%          if((ra(j_row,i_col) <= -cutoff_point_ra_mean) ||
(ra(j_row,i_col) >= cutoff_point_ra_mean))
%          ra(j_row,i_col) = 0;
%         end;
```

```
%     end;
% end;
% for j_row=1:1:257
%     for i_col=1:1:258
% %         if ((ra(j_row,i_col) >= 200) ||  (ra(j_row,i_col) >= 200))
% %             ra(j_row,i_col) = 0;
% %         end;
% %         if ((ra(j_row,i_col) >= 0) &&  (ra(j_row,i_col) <= 200))
% %             ra(j_row,i_col) = 0;
% %         end;
%         if((i_col >= 129) && (i_col <= 500))
%            ra(j_row,i_col) = 0;
%         end;
%     end;
% end;
%
figure(100),%**************************************************************
*************************
% hist(ra),
% title('Histogram: raORIGINAL');
% end of test% end of test% end of test% end of test% end of test% end
of test% end of test% end of test

% 1D wavelet transform at each column of the Radon transform
% -> "Ridgelets".  Care is taken for the non-dyadic size to ensure
% orthonormal condition

if isdyadic_frito_avnpr(p - 1)
    % Number of wavelet decomposition levels
    %     n = log2(p - 1); % 'log2' is a built-in function.
    n=8;
    % Make sure using the periodic extension mode
    st = dwtmode_frito_avnpr('status', 'nodisp');
    if ~strcmp(st, DWTMODE) % 'strcmp' is a built-in function.

    dwtmode_frito_avnpr(DWTMODE);
    end

    % Take wavelet transform at each direction except the last
coefficients
    [r, l] = wavedecc_frito_avnpr(ra(1:end-1, :), n, wname);

    % Incooperate the Radon coefficient to the waveklet approx. coeff
    r(1, :) = (r(1, :) - sqrt(p-1) * ra(end, :)) / sqrt(p); % 'sqrt' is
a built-in function.
% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST%
TEST% TEST% TEST% TEST% TEST% TEST
% r = r.*2;
% end of test% end of test% end of test% end of test% end of test% end
of test% end of test% end of test
else
    error('Have not support this size of image yet!');
end
% toc
```

/***/

process_ri_row.col.m

```
% function
[ri2,row_max1,col_max1,row_mean_geometric,col_mean_geometric] =
process_ri_row_col(ri,row,col)
function
[ri2,row_max1,col_max1,row_mean_geometric,col_mean_geometric,row_mean_s
ampling,col_mean_sampling] =
process_ri_row_col(ri,row,col,flag,TR,TC,G,r,c,r_start,r_end,c_start,c_
end)
Ro_W = row;
Co_L = col;
ri2 = ri;
row_mean_geometric = zeros(Ro_W,1);
col_mean_geometric = zeros(1,Co_L);
row_mean_sampling = zeros(Ro_W,1);
col_mean_sampling = zeros(1,Co_L);

%%
row_max1 = zeros(Ro_W,1);
for j=1:1:Ro_W
        row_max1(j,1) = max(ri2(j,:));
end;
col_max1 = zeros(1,Co_L);
for i=1:1:Co_L
        col_max1(1,i) = max(ri2(:,i));
end;


%%
for j=1:1:Ro_W
    row_mean_geometric(j,1) = (sum(abs(ri2(j,:))))/Co_L;
end;
for i=1:1:Co_L
    col_mean_geometric(1,i) = (sum(abs(ri2(:,i))))/Ro_W;
end;
for j=1:1:Ro_W
    row_mean_sampling(j,1) = (sum(ri2(j,:)))/Co_L;
end;
for i=1:1:Co_L
    col_mean_sampling(1,i) = (sum(ri2(:,i)))/Ro_W;
end;



%%
if(flag == 1)
```

```
   tr = TR;
else
   tr = 0;
end;
if(flag == 1)
   tc = TC;
else
   tc = 0;
end;
if(r==1)
for j=r_start:1:r_end
    for i=c_start:1:c_end
% if(i~=129)
        if(G==1)
            if((abs(ri(j,i))) > (row_mean_geometric(j,1)+tr))
%             if((abs(ri(j,i))) >=
(row_mean_geometric(j,1)+var_ri2(1,i)))

                ri2(j,i) = 0;
%                 ri2(j,i) = ri2(j,i);
            end;
        else
            if((abs(ri(j,i))) < (row_mean_geometric(j,1)+tr))
%             if((abs(ri(j,i))) <=
(row_mean_geometric(j,1)+var_ri2(1,i)))

                ri2(j,i) = 0;
%                 ri2(j,i) = ri2(j,i);
            end;
        end;
% end;
    end;
end;
end;
if(c==1)
for j=r_start:1:r_end
    for i=c_start:1:c_end
% if(i~=130)

        if(G==1)
            if((abs(ri(j,i))) > (col_mean_geometric(1,i)+tc))
%             if((abs(ri(j,i))) >=
(var_ri2(1,i)+col_mean_geometric(1,i)))

                ri2(j,i) = 0;
%                 ri2(j,i) = ri2(j,i);
            end;
        else
            if((abs(ri(j,i))) < (col_mean_geometric(1,i)+tc))
%             if((abs(ri(j,i))) <=
(var_ri2(1,i)+col_mean_geometric(1,i)))
                ri2(j,i) = 0;
%                 ri2(j,i) = ri2(j,i);
            end;
        end;
```

```
% end;
    end;
end;
end;
```

/**/

ifrito_avnpr.m

```
function a = ifrito_avnpr(r, l, m, wname,DWTMODE)
% IFRITO    Inverse orthogonal finite ridgelet transform
%   a = ifrito(r, l, m, wname)
%
% Input:
%       r:      ridgelet coefficients in a (P-1) by (P+1) matrix,
%       one column for each direction
%   l:  structure of the wavelet decomposition that is
%       needed for reconstruction
%   m:  normalized mean value of the image
%   wname:  wavelet name
%
% Output:
%   a:  reconstructed image
%
% See also: FRITO

p = size(r, 2) - 1; % 'size'is a built-in function.
if (size(r, 1) ~= (p - 1)) | ~isprime(p)
    error('Ridgelet coefficients must be in a (P-1) by (P+1) matrix.');
end

% Back to Radon domain by inverting the wavelet transform.
% By definition, Radon coefficients should have zero mean
% at each direction

ra = zeros(p, p + 1); % 'zeros'is a built-in function.

if isdyadic_frito_avnpr(p - 1)
    % Number of wavelet decomposition levels
%     n = log2(p - 1); % 'log2'is a built-in function.
    n=8;
    % Recorrect the wavelet approx. cofficients
    r(1, :) = sqrt(p) * r(1, :) / p; % 'sqrt'is a built-in function.

    % Make sure using the periodic extension mode
    st = dwtmode_frito_avnpr('status', 'nodisp');
    if ~strcmp(st, DWTMODE) % 'strcmp'is a built-in function.
        dwtmode_frito_avnpr(DWTMODE);
    end
```

```
    % Inverse the wavelet transform
    ra(1:end-1, :) = waverecc_ifrito_avnpr(r, l, wname);

    % Compute the last Radon coefficients
    ra(end, :) = -sqrt(p-1) * r(1, :); % 'sqrt'is a built-in function.

else
    error('Have not support this size of image yet!');
end

% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST%
TEST% TEST% TEST% TEST% TEST% TEST
% ra = ra.*(1/2);
% end of test% end of test% end of test% end of test% end of test% end
of test% end of test% end of test

% Inverse the finite Radon transform with the mean corrected
% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST%
TEST% TEST% TEST% TEST% TEST% TEST
% ra = ra.*(2/3);
% ra = ra.*(5/4);
% end of test% end of test% end of test% end of test% end of test% end
of test% end of test% end of test
a = ifrat_ifrito_avnpr(ra, 0);

% Add back the DC component ->>>>>>>>>>> commented by myself
a = a + m / (size(r, 2) - 1); % 'sqrt'is a built-in function.

% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST% TEST%
TEST% TEST% TEST% TEST% TEST% TEST
% a = a.*(2/3);
% end of test% end of test% end of test% end of test% end of test% end
of test% end of test% end of test
/*************************************************************************/
```

calculate_histo.m

```
%% Calculate histogram of imrec_frito2/3
function histogram_calculated = calculate_histo(input_imrec)
%
p = size(input_imrec, 1); % 'size' is a built-in function.
q = size(input_imrec, 2); % 'size' is a built-in function.
%
histogram_calculated = zeros(p,1);
% input_imrec = input_imrec * 255;
% input_imrec = round(input_imrec);
% input_imrec = uint8(input_imrec);

% loop over i ROWS of input image
for i_row_calc_hist_loop=1:1:p
    % loop over j COLS of input image
    for j_col_calc_hist_loop=1:1:q
        k = input_imrec(i_row_calc_hist_loop,j_col_calc_hist_loop);
```

```
        if(k>=0)
            k = round(k);
            k = uint8(k);
            if (k == 0)
                k = 1;
            end;
            histogram_calculated(k) = (histogram_calculated(k) + 1);
        end;
    % end loop over j
    end;
% end loop over i
end;
%
/*********************************************************************/
```

find_cutoff_point.m

```
%       /****************************************
%       *
%       *   find_cutoff_point(..
%       *
%       *   This function looks at a histogram
%       *   and sets a cuttoff point at a given
%       *   percentage of pixels.
%       *   For example, if percent=0.6, you
%       *   start at 0 in the histogram and count
%       *   up until you've hit 60% of the pixels.
%       *   Then you stop and return that pixel
%       *   value.
%       *
%       ****************************************/

function cutoff_value = find_cutoff_point(input, percent, rows, cols)

   double  fsum;
   double  sum_div;
   int    i;

   summ = 0;


   lr      = rows;
   lc      = cols;
   num     = lr*lc;
   fd      = num;

   GRAY_LEVELS = 255;
   cutoff = zeros(256,1);

for j=0:1:255
    cutoff(j+1,1) = j;
end;
```

```
   histogram = calculate_histo(input);

     i = 0;
     looking = 1;

   while looking
      fsum    = summ;
      sum_div = fsum/fd;
      if(sum_div >= percent)
         looking = 0;
      else
          i = i+1;
         summ = summ + histogram(i,:);
      end;
   end; %    /* ends while looking */


   if(i >= (GRAY_LEVELS+1))
       i = GRAY_LEVELS;
   end;
      if(i == 0)
       i = 1;
      end;
   cutoff_value = cutoff(i,1);
% }  /* ends find_cutoff_point */
/*********************************************************************/
```

erode_imrecs.m

```
function output = erode_imrecs(input,value,threshold,SE,rows,cols)
% This function (erode_imrecs) erodes pixels.  If a pixel
% equals erode_value and has more than or equal to erode_threshold
% neighbors equal to 0, then set that
% pixel in the output to 0.
output = input;
countings = 0;
se = SE;

switch(se)

    case 1
            % SE:1-> max_erode_threshold = 8
            %       0   0   0 -> 1st check row wise
            %       0   1   0 -> 2nd check row wise
            %       0   0   0 -> 3rd check row wise
            for i=3:1:rows-3
                for j=3:1:cols-3
                     if(input(i,j) == value)
                        countings = 0;
                        for a=-1:1:1
                            for b=-1:1:1
                                  if(input(i+a,j+b) == 0)
                                     countings = countings+1;
```

```
                        end;
                    end;  %  ends loop over b
                end;  % ends loop over a
               end; % ends if the_image == value */
               if(countings >= threshold)
                   output(i,j) = 0;
               end;  %ends if count>threshold
           end;  % ends loop over j */
       end;  % ends loop over i */

case 2
       % SE:2-> max_erode_threshold = 8
       %       0   0   0 -> 1st check row wise
       %       1   0   0 -> 2nd check row wise
       %       0   0   0 -> 3rd check row wise
       for i=3:1:rows-3
           for j=3:1:cols-3
                if(input(i,j) == value)
                   countings = 0;
                   for a=-1:1:1
                       for b=0:1:2
                             if(input(i+a,j+b) == 0)
                                countings = countings+1;
                             end;
                       end;  %  ends loop over b
                   end;  % ends loop over a
                end; % ends if the_image == value */
                if(countings >= threshold)
                    output(i,j) = 0;
                end;  %ends if count>threshold
           end;  % ends loop over j */
       end;  % ends loop over i */

case 3
       % SE:3-> max_erode_threshold = 8
       %       0   0   0 -> 1st check row wise
       %       0   0   1 -> 2nd check row wise
       %       0   0   0 -> 3rd check row wise
       for i=3:1:rows-3
           for j=3:1:cols-3
                if(input(i,j) == value)
                   countings = 0;
                   for a=-1:1:1
                       for b=-2:1:0
                             if(input(i+a,j+b) == 0)
                                countings = countings+1;
                             end;
                       end;  %  ends loop over b
                   end;  % ends loop over a
                end; % ends if the_image == value */
                if(countings >= threshold)
                    output(i,j) = 0;
                end;  %ends if count>threshold
           end;  % ends loop over j */
       end;  % ends loop over i */
```

```
case 4
        % SE:4-> max_erode_threshold = 8
        %       0   0   0 -> 1st check row wise
        %       0   0   0 -> 2nd check row wise
        %       0   0   1 -> 3rd check row wise
        for i=3:1:rows-3
            for j=3:1:cols-3
                 if(input(i,j) == value)
                    countings = 0;
                    for a=-2:1:0
                        for b=-2:1:0
                              if(input(i+a,j+b) == 0)
                                 countings = countings+1;
                              end;
                        end;  %  ends loop over b
                    end;  % ends loop over a
                 end; % ends if the_image == value */
                 if(countings >= threshold)
                     output(i,j) = 0;
                 end;  %ends if count>threshold
            end;  % ends loop over j */
        end;  % ends loop over i */

case 5
        % SE:5-> max_erode_threshold = 24
        %       0           0           0
        %           0       0       0
        %               0   0   0
        %       0   0   0   1   0   0   0
        %               0   0   0
        %           0       0       0
        %       0           0           0
        for i=3:1:rows-3
            for j=3:1:cols-3
                 if(input(i,j) == value)
                    countings = 0;
                    for a=-3:1:3
                        if(abs(a) == 3)
                            for b=-3:3:3
                                  if(input(i+a,j+b) == 0)
                                     countings = countings+1;
                                  end;
                            end;  %  ends loop over b
                        end;
                        if(abs(a) == 2)
                            for b=-2:2:2
                                  if(input(i+a,j+b) == 0)
                                     countings = countings+1;
                                  end;
                            end;  %  ends loop over b
                        end;
                        if(abs(a) == 1)
                            for b=-1:1:1
                                  if(input(i+a,j+b) == 0)
```

```
                                countings = countings+1;
                            end;
                        end;  %  ends loop over b
                    end;
                    if(a == 0)
                        for b=-3:1:3
                            if(input(i+a,j+b) == 0)
                                countings = countings+1;
                            end;
                        end;  %  ends loop over b
                    end;
                end;  % ends loop over a
            end; % ends if the_image == value */
            if(countings >= threshold)
                output(i,j) = 0;
            end;  %ends if count>threshold
        end;  % ends loop over j */
    end;  % ends loop over i */

case 6
    % SE:6-> max_erode_threshold = 40
    %       0   0   0   0   0   0   0
    %       0   0       0       0   0
    %       0       0   0   0       0
    %       0   0   0   1   0   0   0
    %       0       0   0   0       0
    %       0   0       0       0   0
    %       0   0   0   0   0   0   0
    for i=4:1:rows-4
        for j=4:1:cols-4
            if(input(i,j) == value)
                countings = 0;
                for a=-3:1:3
                    if(abs(a) == 3)
                        for b=-3:1:3
                            if(input(i+a,j+b) == 0)
                                countings = countings+1;
                            end;
                        end;  %  ends loop over b
                    end;
                    if(abs(a) == 2)
                        for b=-2:2:2
                            if(input(i+a,j+b) == 0)
                                countings = countings+1;
                            end;
                        end;  %  ends loop over b
                        for b=-3:6:3
                            if(input(i+a,j+b) == 0)
                                countings = countings+1;
                            end;
                        end;  %  ends loop over b

                    end;
                    if(abs(a) == 1)
                        for b=-1:1:1
```

```
                                    if(input(i+a,j+b) == 0)
                                       countings = countings+1;
                                    end;
                                end;  %  ends loop over b
                                for b=-3:6:3
                                    if(input(i+a,j+b) == 0)
                                       countings = countings+1;
                                    end;
                                end;  %  ends loop over b
                            end;
                            if(a == 0)
                                for b=-3:1:3
                                    if(input(i+a,j+b) == 0)
                                       countings = countings+1;
                                    end;
                                end;  %  ends loop over b
                            end;
                        end;  % ends loop over a
                    end; % ends if the_image == value */
                    if(countings >= threshold)
                        output(i,j) = 0;
                    end;  %ends if count>threshold
                end;  % ends loop over j */
            end;  % ends loop over i */

    case 7
            % SE:7-> max_erode_threshold = 17
            %            0       0       0
            %                0   0   0
            %            0   0   1   0   0
            %                0   0   0
            %            0       0       0
            for i=4:1:rows-4
                for j=4:1:cols-4
                     if(input(i,j) == value)
                        countings = 0;
                        for a=-2:1:2
                            if(abs(a) == 2)
                                for b=-2:2:2
                                    if(input(i+a,j+b) == 0)
                                       countings = countings+1;
                                    end;
                                end;  %  ends loop over b
                            end;
                            if(abs(a) == 1)
                                for b=-1:1:1
                                    if(input(i+a,j+b) == 0)
                                       countings = countings+1;
                                    end;
                                end;  %  ends loop over b
                            end;
                            if(a == 0)
                                for b=-3:1:3
                                    if(input(i+a,j+b) == 0)
                                       countings = countings+1;
```

```
                                end;
                            end;  %  ends loop over b
                        end;
                    end;  % ends loop over a
                end; % ends if the_image == value */
                if(countings >= threshold)
                    output(i,j) = 0;
                end;  %ends if count>threshold
            end;  % ends loop over j */
        end;  % ends loop over i */


case 8
        % SE:8-> 9x9->max_erode_threshold = 80
        %       0   0   0 -> 1st check row wise
        %       0   1   0 -> 2nd check row wise
        %       0   0   0 -> 3rd check row wise
        for i=5:1:rows-5
            for j=5:1:cols-5
                if(input(i,j) == value)
                   countings = 0;
                   for a=-4:1:4
                       for b=-4:1:4
                             if(input(i+a,j+b) == 0)
                                countings = countings+1;
                             end;
                       end;  %  ends loop over b
                   end;  % ends loop over a
                end; % ends if the_image == value */
                if(countings >= threshold)
                    output(i,j) = 0;
                end;  %ends if count>threshold
            end;  % ends loop over j */
        end;  % ends loop over i */

case 9
        % SE:9-> max_erode_threshold = 9
        %           1   0   0   0   0   0   0   0   0   0
        for i=1:1:rows-4
            for j=1:1:cols-10
                if(input(i,j) == value)
                   countings = 0;
                       for b=0:1:9
                             if(input(i,j+b) == 0)
                                countings = countings+1;
                             end;
                       end;  %  ends loop over b
                end; % ends if the_image == value */
                if(countings >= threshold)
                    output(i,j) = 0;
                end;  %ends if count>threshold
            end;  % ends loop over j */
        end;  % ends loop over i */

case 10
```

```
        % SE:10-> max_erode_threshold = 10
        %           0   0   0   0   0   1   0   0   0   0   0
        for i=1:1:rows
            for j=6:1:cols-6
                 if(input(i,j) == value)
                    countings = 0;
                        for b=-5:1:5
                              if(input(i,j+b) == 0)
                                 countings = countings+1;
                              end;
                        end;  %  ends loop over b
                 end; % ends if the_image == value */
                 if(countings >= threshold)
                     output(i,j) = 0;
                 end;  %ends if count>threshold
            end;  % ends loop over j */
        end;  % ends loop over i */

case 11
        % SE:11-> max_erode_threshold = 10
        %           (0   0   0   0   0   1   0   0   0   0   0)'
        for i=1:1:cols
            for j=6:1:rows-6
                 if(input(j,i) == value)
                    countings = 0;
                        for b=-5:1:5
                              if(input(j+b,i) == 0)
                                 countings = countings+1;
                              end;
                        end;  %  ends loop over b
                 end; % ends if the_image == value */
                 if(countings >= threshold)
                     output(j,i) = 0;
                 end;  %ends if count>threshold
            end;  % ends loop over j */
        end;  % ends loop over i */

case 12
        % SE:12-> max_erode_threshold = 20
        %           (0   0   0   0   0   1   0   0   0   0   0)'
        for i=1:1:cols
            for j=11:1:rows-11
                 if(input(j,i) == value)
                    countings = 0;
                        for b=-10:1:10
                              if(input(j+b,i) == 0)
                                 countings = countings+1;
                              end;
                        end;  %  ends loop over b
                 end; % ends if the_image == value */
                 if(countings >= threshold)
                     output(j,i) = 0;
                 end;  %ends if count>threshold
            end;  % ends loop over j */
        end;  % ends loop over i */
```

```
case 14
    % SE = max_erode_threshold = 75
    %5x15
            for i=3:1:rows-3
                for j=8:1:cols-8
                     if(input(i,j) == value)
                        countings = 0;
                        for a=-2:1:2
                            for b=-7:1:7
                                  if(input(i+a,j+b) == 0)
                                     countings = countings+1;
                                  end;
                            end;  %  ends loop over b
                        end;  % ends loop over a
                     end; % ends if the_image == value */
                     if(countings >= threshold)
                         output(i,j) = 0;
                     end;  %ends if count>threshold
                end;  % ends loop over j */
            end;  % ends loop over i */

case 15
    % SE = max_erode_threshold = 147
    %7x21
            for i=4:1:rows-4
                for j=11:1:cols-11
                     if(input(i,j) == value)
                        countings = 0;
                        for a=-3:1:3
                            for b=-10:1:10
                                  if(input(i+a,j+b) == 0)
                                     countings = countings+1;
                                  end;
                            end;  %  ends loop over b
                        end;  % ends loop over a
                     end; % ends if the_image == value */
                     if(countings >= threshold)
                         output(i,j) = 0;
                     end;  %ends if count>threshold
                end;  % ends loop over j */
            end;  % ends loop over i */

case 16
        % SE:16-> max_erode_threshold = 20
        %           (0   0   0   0   0   1   0   0   0   0   0)'
        for i=1:1:cols
            for j=11:1:rows-11
                 if(input(j,i) == value)
                    countings = 0;
                        for b=-10:1:10
                              if(input(j+b,i) == 0)
                                 countings = countings+1;
                              end;
                        end;  %  ends loop over b
```

```
                    end; % ends if the_image == value */
                    if(countings >= threshold)
                        output(j,i) = 0;
                    end;  %ends if count>threshold
                end;  % ends loop over j */
            end;  % ends loop over i */

    case 17
            % SE:17-> max_erode_threshold = 20
            %           (0   0   0   0   0   1   0   0   0   0   0)' R
            for i=4:1:cols-4
                for j=1:1:rows
                    if(input(j,i) == value)
                       countings = 0;
                           for b=-3:1:3
                                 if(input(j,i+b) == 0)
                                    countings = countings+1;
                                 end;
                           end;  %  ends loop over b
                    end; % ends if the_image == value */
                    if(countings >= threshold)
                        output(j,i) = 0;
                    end;  %ends if count>threshold
                end;  % ends loop over j */
            end;  % ends loop over i */

    otherwise
            % SE:1-> max_erode_threshold = 8
            %       0   0   0 -> 1st check row wise
            %       0   1   0 -> 2nd check row wise
            %       0   0   0 -> 3rd check row wise
            for i=11:1:rows-11
                for j=11:1:cols-11
                    if(input(i,j) == value)
                       countings = 0;
                       for a=-1:1:1
                           for b=-1:1:1
                                 if(input(i+a,j+b) == 0)
                                    countings = countings+1;
                                 end;
                           end;  %  ends loop over b
                       end;  % ends loop over a
                     end; % ends if the_image == value */
                     if(countings >= threshold)
                         output(i,j) = 0;
                     end;  %ends if count>threshold
                end;  % ends loop over j */
            end;  % ends loop over i */

end;
/*********************************************************************/
```