GAZİANTEP UNIVERSITY GRADUATE
SCHOOL OF NATURAL & APPLIED SCIENCES

AUTOMATIC VEHICLE IDENTIFICATION BY PLATE RECOGNITION

M. Sc. THESIS
IN
ELECTRICAL & ELECTRONICS ENGINEERING

BY
SERKAN ÖZBAY
JANUARY 2006

AUTOMATIC VEHICLE IDENTIFICATION BY PLATE RECOGNITION

M.Sc. Thesis
in
Electrical & Electronics Engineering
University of Gaziantep

Supervisor
Assoc. Prof. Ergun ERÇELEBİ

by

Serkan ÖZBAY

January 2006

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Sadettin ÖZYAZICI

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Assoc. Prof. Gülay TOHUMOĞLU

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Ergun ERÇELEBİ

Supervisor

Examining Committee Members

1. Prof. Dr. Rauf MİRZABABAYEV	…………………..
2. Prof. Dr. Arif NACAROĞLU	…………………..
3. Assoc. Prof. Ergun ERÇELEBİ	…………………..
4. Assist. Prof. Tuncay YAŞAR	…………………..
5. Assist. Prof. Nurdal VATSUJİ	..………………...

ABSTRACT

AUTOMATIC VEHICLE IDENTIFICATION BY PLATE RECOGNITION

ÖZBAY, Serkan
M.Sc. in Electrical & Electronics Eng.
Supervisor: Assoc. Prof. Ergun ERÇELEBİ
January 2006, 115 pages

In this thesis, a new algorithm for automatic vehicle identification has been proposed. Identification is made by license plate recognition (LPR) algorithm. License plate recognition is a form of automatic vehicle identification and is an algorithm that identifies the vehicle by recognizing the license plate automatically. In this study, a simple but effective algorithm is presented for vehicle's license plate recognition system.

There are some other methods to identify the vehicles such as bar code-based identification systems, radio-frequency identification systems. In bar code-based identification and radio-frequency identification systems, the methods require external components installed on vehicles for automated identification. On the other hand, license plate recognition technology identifies the vehicle using only its license plate. Since every vehicle carries a unique license plate, no external cards, tags or transmitters need to be recognizable. Due to this reason, license plate recognition algorithm is the most powerful and useful technique for automatic vehicle identification.

The proposed algorithm consists of three major parts: Extraction of plate region, segmentation of plate characters and recognition of plate characters. For extracting the plate region, edge detection algorithms and smearing algorithms are used. In segmentation part, smearing algorithms, filtering and some morphological algorithms are used. And finally statistical based template matching is used for recognition of plate characters.

This algorithm operates on inactive real images and the system is designed for the identification of Turkish license plates. The necessary codes for the proposed algorithm were written in Matlab software. To see the overall performance, the proposed algorithm has been tested over a large number of images. The images were taken on different time periods of the day and also these test images were taken under various illumination conditions. And it was obtained that %97.6 success rate for the extraction of plate region, %96 success rate for the segmentation of the characters and %98.8 accuracy rate for the recognition of plate characters, giving the overall system performance as %92.57 recognition rate.

Moreover, some image processing techniques that have been used for license plate recognition were presented in this thesis. And the advantages and the use of these techniques have been explained.

Key Words: Character recognizer, license plate recognition, plate region extraction, segmentation, smearing, template matching

ÖZET

PLAKA TANIMA YÖNTEMİYLE OTOMATİK ARAÇ TESPİTİ

ÖZBAY Serkan

Yüksek Lisans Tezi, Elektrik-Elektronik Müh. Bölümü
Tez Yöneticisi: Doç.Dr. Ergun ERÇELEBİ
Ocak 2006, 115 sayfa

Bu tezde, otomatik araç tespitinde kullanılan yeni bir algoritma önerilmektedir. Araç tespiti, plaka tanıma algoritması ile yapılır. Plaka tanıma algoritması otomatik araç tespitinde kullanılan yöntemlerden bir tanesidir ve araçları, plakalarını otomatik olarak tanıyarak tespit eden bir algoritmadır. Bu çalışmada, basit fakat etkili bir tanıma algoritması sunulmuştur.

Araçların tespiti için barkod tabanlı veya radyo frekansı ile çalışan daha farklı metodlar vardır. Bu metodların kullanıldığı sistemlerde, otomatik araç tespiti için araçlara yerleştirilmiş harici bir elemana ihtiyaç vardır. Bununla birlikte, plaka tanıma algoritması aracı sadece plakasını kullanarak tespit eder. Her araç kendine ait özel bir plaka taşıdığı için, tanınma için harici bir elemana, ek bir parçaya yada vericiye ihtiyaç yoktur. Bu nedenden dolayı plaka tanıma algoritması, araçların otomatik tespiti için en etkili ve kullanışlı tekniktir.

Çalışmada önerilen algoritma üç temel bölümden oluşmaktadır : Plaka bölgesinin çıkartılması, plaka karakterlerin ayrıştırılması ve plaka karakterlerinin tanınması. Plaka bölgesi çıkartılırken, kenar belirleme, lekeleme algoritmaları kullanılmıştır. Ayrıştırma bölümünde, lekeleme, filtreleme ve bazı morfolojik algoritmalar kullanılmaktadır. Ve plaka karakterlerinin tanınması için istatiksel temellli şablon eşleştirme kullanılmıştır.

Bu sistem duran araç görüntüleri üzerinde çalışmaktadır ve sistem Türk araç plakalarının tespiti için tasarlanmıştır. Önerilen algoritma için gerekli olan kodlar Matlab programı ile yazılmıştır. Sistemin performansını tam olarak görebilmek için, önerilen algoritma çok miktarda görüntü ile test edilmiştir. Kullanılan görüntüler günün değişik zaman periyotlarında ve farklı aydınlanma durumlarında çekilmiştir. Ve plaka bölgesinin çıkartılmasında %97.6 başarı oranı, plaka karakterlerin ayrıştırılması için %96 ve plaka karakterlerinin tanınmasında %98.8 doğruluk oranı, ve sistemin bütünü için %92.57 tanıma oranı elde edilmiştir.

Bu tezde ayrıca plaka tanıma algoritmasında kullanılan bazı görüntü işleme teknikleri sunulmuştur ve bu tekniklerin avantajları ve kullanımları açıklanmıştır.

Anahtar Kelimeler: Karakter tanıyıcı, plaka tanıma, plaka bölgesi çıkarma, ayrıştırma, lekeleme, şablon eşleştirme

ACKNOWLEDGMENTS

I express sincere appreciation to my supervisor, Assoc. Prof. Ergun ERCELEBI, for his guidance, constructive proofreading, and many fruitful discussions. I thank him for helping me during writing this thesis.

I wish also to acknowledge my parents for always supporting my choices in life and my colleague at GMYO for their support and encouragement.

I would like to express my gratitude to my teachers at the Department of Electrical and Electronics Engineering.

And finally, special thanks to my fiancee, Gulden AKAY, for sharing the most valuable times with me.

CONTENTS

LIST OF FIGURES page

LIST OF TABLES

LIST OF SYMBOLS

$a(x, y)$	Two dimensional light intensity function
D	Dilation operation
E	Erosion operation
O	Opening operation
C	Closing operation
∇	Gradient of a function
$R(m, n)$	Cross correlation function
A	Recognition rate

CHAPTER 1

INTRODUCTION

With the rapid development of public transportation systems and increasing use of vehicles in recent years, automatic identification of vehicles has played an important role in many applications. For these reasons, researchers tend to the research on advanced electronic and computer vision technologies to monitor and control the traffic.

There have been some methods to identify the vehicles such as bar code-based identification systems, radio-frequency identification systems and license plate recognition systems. In bar code-based identification and radio-frequency identification systems, the methods require external components installed on vehicles for automated identification. On the other hand, license plate recognition technology identifies the vehicle using only its license plate. Since every vehicle carries a unique license plate, no external cards, tags or transmitters need to be recognizable. This important advantage has increased the use of vision-based recognition systems.

License Plate Recognition (LPR) is a form of Automatic Vehicle Identification (AVI) based on automatic recognition of vehicle's license plates. This plate recognition technology has a wide range of applications which may solve numerous tasks related to the identification of cars such as:

- Automated parking attendance at garages and parking lots
- Border crossing control
- Identification of stolen cars
- Speed enforcement
- Red light violation enforcement

- Electronic toll collection
- Security control

License Plate Recognition (LPR) algorithm may have different names or it may be called with slightly different names at various studies and references given as:

- Automatic Vehicle Identification (AVI)
- Car Plate Recognition (CPR)
- Automatic Number Plate Recognition (ANPR)
- Car Plate Reader (CPR)
- Optical Character Recognition (OCR) for cars

A typical license plate recognition system is shown in Figure 1.1.

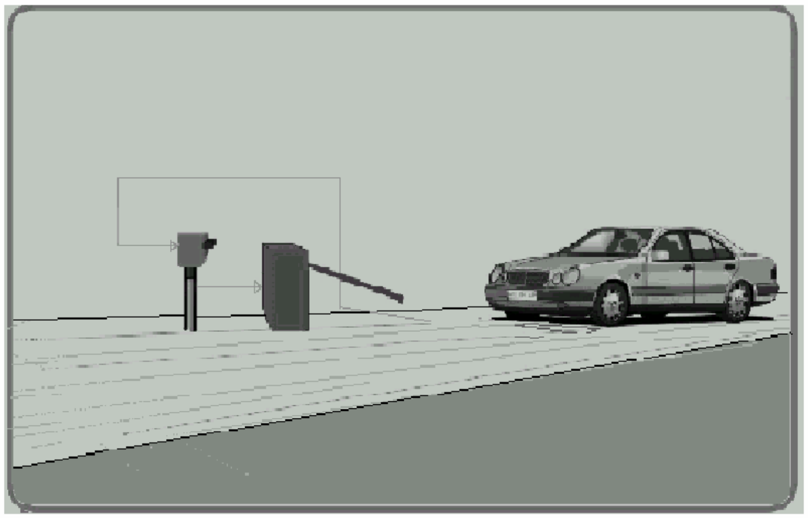


Figure 1.1: A typical scheme for the LPR system

Plate recognition system uses a video camera that captures a frame when the sensors detect a vehicle. The captured image frame is sent to the computer using an interface between the camera and the computer. Taken image frame is loaded to the memory of the computer and then computer runs the license plate recognition application which controls the system, reads the images, analyzes and the identifies the plate. The main part of the system is the software which analyzes the image and extracts the plate information. For analyzing the image and identifying the license plate of the vehicle, some image processing techniques are applied on the image using the software.

Some LPR systems operate on the successive frames taken by the camera as well as some operates on a single frame. Working on the successive frames has the advantage that the exterior edge of the vehicle is detected using the motion detection algorithm [1]. The method of detecting cars uses the rate of color change in vertical sensor. Vertical sensor is a special region of image. The difference value between the vertical sensors of continuous images show whether there is a car or not. If difference value is higher than the threshold value, it is regarded that the current image contains car [2]. Working on the successive frames has also the advantage that noise on the image can be eliminated by taking the mean value of the successive plate images. However, if the number of images increases, it needs more memory requirements. This is the disadvantage of that method.

In this study, license plate recognition algorithm consists of three major units:

- Extraction of Plate Region
- Segmentation of Plate Characters
- Recognition of Plate Characters

Extraction of plate region is the algorithm for finding the location of plate on vehicle image. Segmentation of plate characters is to separate the plate characters on a plate individually. Finally, recognition of characters is to identify the plate characters correctly.

In this work, the images for the input to the system are colored images with the size 1200x1600x3. The captured image is taken from 4-5 meters away from the car. The input image is first converted to the binary image consisting of only 1's and 0's (only black and white). Then some image processing techniques are applied to the binary image for extraction of plate location such as smearing algorithm, filtering and edge detection algorithms. After extracting the plate region, license plate is segmented into its constituent parts obtaining the characters individually by segmentation unit. In segmentation part, smearing algorithm, filtering and morphological operations are used. And finally recognition unit identifies the characters giving the result as the plate number. For the recognition, statistical based template matching is used. For the realization of this system, Matlab 6.5 is used and the necessary algorithm is designed and coded in Matlab. This system is designed for the recognition of Turkish license plates.

The thesis is organized as follows. Chapter 2 presents the history of Automatic vehicle identification systems in literature. In chapter 3, the image processing algorithms used for license plate recognition are discussed involving previously worked algorithms and the proposed algorithm. Extraction of plate region is explained in chapter 4. In this chapter, segmentation of plate characters and the details of recognition of characters are also presented. In chapter 5, the experimental results of the proposed algorithm are discussed. And finally, conclusion is made and the future studies are given in the last chapter.

CHAPTER 2

HISTORY OF AVI SYSTEMS & LITERATURE SURVEY

2.1 History of AVI Systems

The name “Automatic Vehicle Identification (AVI)” is used for all technologies related with the identification or recognition of particular vehicle. Early development of AVI systems occurred in the United States, beginning with an optical scanning system in the 1960s to identify railroad box cars. Since then, there have been enormous advances in microelectronics. Inductive loop, radio frequency, infrared, and microwave systems have all been developed and even satellites can be used to provide continuous monitoring of vehicles. These technological advances and increased accuracy and reliability, along with rapidly diminishing costs, have opened new options for use [3].

As cities became increasingly congested with road traffic in the 1950s and 1960s, the response from developed countries was to build vast freeways. The ultimate failure of that response to overcome traffic problems, along with growing environmental awareness, led to consideration of alternative ways of managing traffic. Traffic planners focused on ways of restraining traffic. If they could persuade, encourage, or force drivers to reduce their road use or change their patterns of use, then existing roads could be used more efficiently. This would reduce traveling time and fuel costs as well as air, noise, and visual pollution.

Planners and policy makers have considered a number of measures to achieve such restraint: taxes on ownership and registration of cars to reduce the number of people able to afford cars; physical barriers to prevent people from driving into congested city areas, or parking controls to discourage them; taxes on fuel as a mean of indirectly charging for road use. Singapore introduced an area licensing system in 1975 based on manual collection of permit fees, which reduced peak hour traffic in the central business district by 40%, but few planners thought that this system could be easily applied to other countries with less authoritarian styles of government [4].

Developments in automatic vehicle identification and monitoring technology opened the possibility for a system of road pricing in which charges depend on the time of day, the road, and the vehicle. This would allow, for example, higher charges to be made for travel during rush hours and on specific congested roads. According to neoclassical economics, this would improve the efficiency with which the roads are used.

The first major trial of the technology for electronic road pricing was undertaken in Hong Kong between 1983 and 1985. A volunteer vehicle fleet was fitted with electronic number plates. Loops beneath the road surface transmitted back to a control center the unique identification number of the passing vehicle. Vehicles which failed to respond with a valid or operative electronic number plate were photographed by closed-circuit TV; this back-up system combined with the basic AVI system ensured a high degree of overall accuracy [5].

Research and development on AVI continues, especially in Europe and Japan [6]. "Prometheus" is the name of an eight-year, $900 million industry-funded program involving six European countries. Its aim is to improve safety, economy, efficiency, comfort, and reduce pollution through the development of an intelligent vehicle [7]. A smaller program called DRIVE is jointly funded by governments and corporations. It is concerned with AVI technologies, computers in vehicles, "smart cards" (that can automatically register electronic transactions), and automatic enforcement systems including cameras and license plate systems.

In Japan, there is a large program organized under MITI called the Intelligent Vehicle System, similar in size and orientation to Prometheus but with more emphasis on artificial intelligence and automatic chauffeur. Another Japanese venture is AMTICS (Advanced Mobile Traffic Information and Communication

System), an integrated traffic information and navigation system which combines CD-ROM technology with AVI [8]. Some Japanese cars on the market come with rudimentary autonomous route guidance systems. The Japanese government has halted the large programs and brought the project leaders together to decide future directions [9].

In the Netherlands, there was an ambitious plan to introduce a national electronic road pricing system by the early 1990s, in which each vehicle would hold a smart card whose balance would be electronically decremented on passing beacons, with a back-up monitoring and enforcement system for vehicles without a valid card number or without sufficient funds. But later this system has been postponed [10].

In Norway, several towns have adopted AVI toll systems. For example, in Alesund there is a programmable remote identification ("premid") system for toll collection on a recently completed island-linking tunnel and bridge system. Sensors in the road alert the system of an approaching car. Antennae send out a weak microwave signal which is reflected back from a identifier plate on the car. This is analyzed by the premid computer to identify the car, confirm that it is a paying subscriber and register the trip. This is recorded in 150 milliseconds. The information collected automatically on the site is then sent to a central computer. Cars that attempt to pass through without paying activate a video camera which records the registration number and the time and place [11].

As a conclusion, the developments on Automatic Vehicle Identification systems are well-developed in USA.

2.2 Literature Survey

In the recent past, researchers have tested a wide array of technologies in an attempt to find improved methods of monitoring traffic conditions. Those techniques can be grouped into roadside techniques, vehicle techniques and license plate recognition techniques. Roadside techniques use detecting devices physically located along the study routes whereas vehicle techniques use detecting devices carried inside vehicles. On the other hand, license plate recognition techniques need no external device or equipments. AVI system comprises one of those advanced technologies currently be used. A brief survey of technologies explored during the past decade to provide an understanding of the level of research interest in traffic surveillance technologies.

Bohnke and Pfannerstill [12] introduced a pattern recognition algorithm, which could utilize unique vehicle presence signatures generated by successive series of inductance loop detectors system. By identifying and reidentifying platoons of vehicles traveling across links bounded by loop detection equipment, vehicle travel times could be determined. Ju and Maze [13] performed simulations on incident detection strategies using the FREQ8PE simulation model. Their research evaluated a comparison of incident detection strategies using police patrol versus the use of motorist call boxes at 1-km spacing. The motorist call boxes formed the backbone of the modeled freeway surveillance and control system (FSCS). This FSCS yielded a benefit-to-cost ratio of 2.69 as it generated benefits from travel-time reduction and reduced fuel consumption. These benefits were brought about by reduced incident detection time afforded by the motorist call boxes. In the development of video-based surveillance, Berka and Lall [14] claim that loop detection reliability is low, and that maintenance and repair of such a pavement-based system creates safety risks for repair crews. Berka and Lall maintain that non-intrusive technologies such as video surveillance provide reduced traffic disruption during installation or repair. In addition, video surveillance is capable of detecting incidents on the sides of roadways, outside of the detection range of loop detectors.

Automatic vehicle identification (AVI) represents a major technological advance in the traffic surveillance technology (Bergan, et. al,) [15]. It origins from the railway industry to monitor the movement of trains, to enable efficient scheduling, and more importantly, to reduce potential conflicts or collisions. Prior to the installation of an AVI system in Houston, Texas, there already had several AVI system existed, which including Hong Kong Electronic Road-Pricing Project (1983); San Francisco International Airport toll revenue collection (1985); Singapore Road-Pricing study (1986); Heavy vehicle Electronic License Plate (HELP) program (1991) etc. All those projects suggest that the implementation of accurate, dependable AVI system is currently possible and the use of AVI systems has the great potential to provide significant monetary saving.

In 1991, a cellular phone demonstration project was designed to monitor freeway traffic conditions in north Houston as a test of Houston AVI system. Researchers recruited 200 volunteers to participate in the program, which required them to call a traffic information office when they passed specific freeway locations during their morning and evening commutes. The lessons learned from the cell phone project aided in the development of the data analysis, processing and dissemination techniques used for the AVI system that was later constructed in Houston and San Antonio. In a similar scenario, prior to installing a large-scale AVI system in the Puget Sound area, a small-scale test of AVI was performed (Butterfield et. al) [16]. In this test, AVI was "piggy-backed" with existing loop detectors. Results yielded an AVI detection rate of about 80% for a fleet of tag-equipped buses. In a 1996 report by Turner [17], a variety of techniques for travel time data collection were discussed, along with the advantages and disadvantages of each. These data collection techniques included electronic distance measuring instruments (DMI's), License plates matching, Cellular phone tracking, Automatic vehicle location (AVL), Automatic vehicle identification (AVI) and Video imaging. Turner specifically noted that travel time information was of particular importance for applications including congestion measurement and real-time travel information.

And the other method, license plate recognition algorithm, for automatic vehicle identification is based on the recognition of cars using their license plates only. After several decades of research, many advances have been achieved in the area of character recognition approaches including artificial neural network [18], learning vector quantization [19] and support vector networks [20]. As it is known, the performance of a character recognition system is based significantly on the recognition feature used. Now recognition features fall into two main categories: Structural features and frequency features. Structural features can precisely describe the structure of a character and success to be used in the recognition of handwritten characters. But they are vulnerable to the recognition of low resolution gray characters such as video index or characters in vehicle license plate. It is difficult to extract invariability structural features because of deform and variation existing in low resolution gray characters [21].

Another feature extraction method is frequency feature extraction method such as Fourier Transform [22] and Wavelet Transform [23]. These methods are widely used for the recognition of low resolution gray character. Gabor Filter, a kind of frequency filter, which has been applied to texture analysis [24], moving object tracking [25], face recognition and character recognition field. Daugman [26] discovered that simply cells in the visual cortex can be modeled by Gabor filter. The 2-D Gabor filters proposed by Daugman are local spatial filter that conjoin information in the 2-D spatial and 2-D Fourier domains. Gabor filter performs a spatial frequency analysis on image. Tavsanoglu and Saatci [27] proposed an approach to form orientation map as recognition feature using a Gabor filter for recognizing characters. Yoshimura and Etoh [28] used Gabor jets projection to form a feature vector for recognizing low resolution gray-scale character.

Lotufo, Morgan and Johnson [29] proposed automatic number-plate recognition using optical character recognition techniques. Johnson and Bird [30] proposed knowledge-guieded boundary following and template matching for automatic vehicle identification. Fahmy [31] discussed about the potential of using the bidirectional associative memories (BAM) neural network for number plate reading. It’s appropriate for small numbers of patterns. Nijhuis, Ter Brugge, Helmholf J.P.W. Pluim, L. Spaanenburg, R.S. Venema and M.A.Westenberg [32] proposed fuzzy logic and neural networks for car LPR. This method used fuzzy logic for segmentation and discrete-time cellular neural networks (DTCNN’S) for feature extraction.

Choi [33] and Kim [34] proposed the method based on vertical edge using Hough Transform (HT) for extracting the license plate. This is assumed that only a license plate has vertical edges in front image of a car. However, many car images have vertical edges from a radiator, and also using Hough Transform (HT) is very sensitive to deformation of plate boundaries. Moreover, this method needs much processing time. E.R. Lee, P.K. Kim and H.J. Kim [35] used neural network for color extraction and a template matching to recognize characters. S.K. Kim, D.W. Kim and

H.J. Kim [36] used a genetic algorithm based segmentation to extract the plate region. Hontani et.al. [37] proposed a method for extracting characters without prior knowledge of their position and size in the image. Park et. al. [38] devised a method to extract Korean license plate depending on the color of the plate. H.J. Kim, D.W. Kim, S.K. Kim, J.V. Lee, J.K. Lee [39] proposed a method of extracting plate region based on color image segmentation by distributed genetic.

CHAPTER 3

IMAGE PROCESSING BASICS FOR LICENSE PLATE RECOGNITION

Interest in digital image processing methods stems from two principal areas: improvement of pictorial information for human interpretation, and processing of scene data for autonomous machine perception.

This chapter presents the fundamentals of image processing techniques and algorithms used for license plate recognition algorithms starting from digital image definition to pre-processing techniques, image enhancement techniques, and image analysis methods.

3.1 Digital Image Fundamentals

We begin with certain basic definitions. An “image” refers to a two-dimensional (2D) light intensity function a(x, y), where x and y denote spatial coordinates and the value of a at any point (x, y) is proportional to the brightness (or gray level) of the image at that point.

A digital image a[m, n] described in a 2D discrete space is derived from an analog image a(x,y) in a 2D continuous space through a sampling process that is frequently referred to as digitization.

The 2D continuous image a(x, y) is divided into N rows and M columns. A digital image that has been discretized both in spatial coordinates and in brightness may be considered as a matrix whose row and column indices identify a point in the image and the corresponding matrix element value identifies the gray level at that point. The intersection of a row and a column is termed a “pixel”. The value assigned to the integer coordinates [m,n] with {m=0,1,2,...,M-1} and {n=0,1,2,...,N-1} is a[m,n].

The image shown in Figure 3.1 has been divided into N = 16 rows and M = 16 columns. The value assigned to every pixel is the average brightness in the pixel rounded to the nearest integer value. The process of representing the amplitude of the 2D signal at a given coordinate as an integer value with L different gray levels is usually referred to as amplitude quantization or simply quantization.

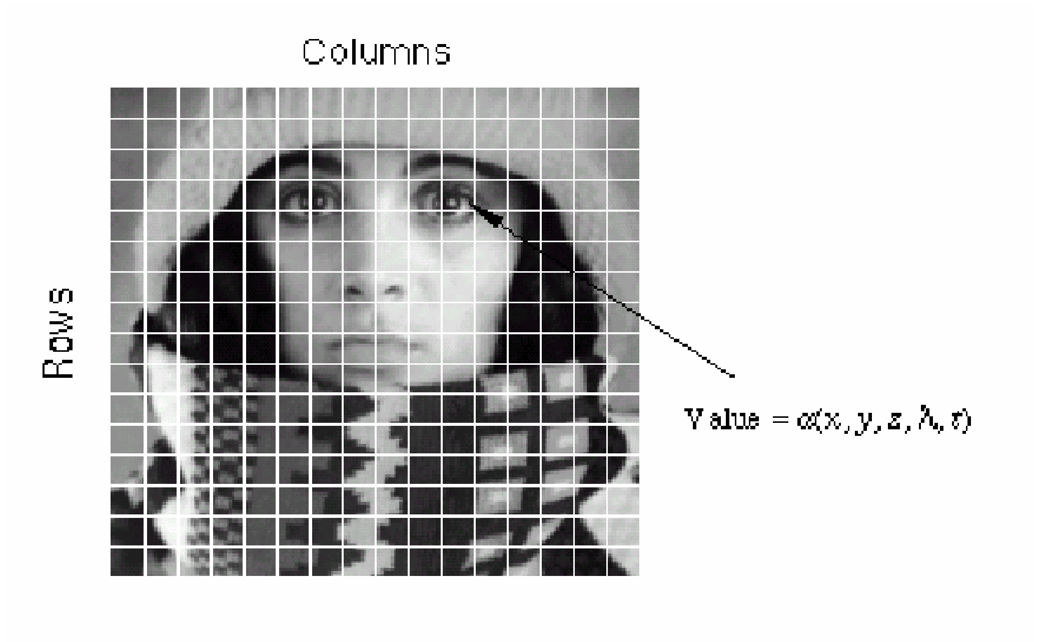


Figure 3.1: Digitization of a continuous image.

There are standard values for the various parameters encountered in digital image processing. These values can be caused by video standards, by algorithmic requirements, or by the desire to keep digital circuitry simple. Table 3.1 gives some commonly encountered values.

Table 3.1: Common values of digital image parameters

Parameter	Symbol	Typical Values
Rows	N	256,512,525,625,1024,1035
Columns	M	256,512,768,1024,1320
Gray Levels	L	2,64,256,1024,4096,16384

The number of distinct gray levels is usually a power of 2, that is, L=2B where B is the number of bits in the binary representation of the brightness levels. When B>1 we speak of a gray-level image; when B=1 we speak of a binary image. In a binary image there are just two gray levels which can be referred to, for example, as "black" and "white" or "0" and "1" [40].

3.2 Characteristics of Image Operations

The types of operations that can be applied to digital images to transform an input image a[m,n] into an output image b[m,n] (or another representation) can be classified into three categories as shown in Table 3.2.

Table 3.2: Types of image operations.

Operation	Characterization
Point	the output value at a specific coordinate is dependent only on the input value at that same coordinate.
Local	the output value at a specific coordinate is dependent on the input values in the neighborhood of that same coordinate.
Global	the output value at a specific coordinate is dependent on all the values in the input image.

This is shown graphically in Figure 3.2.

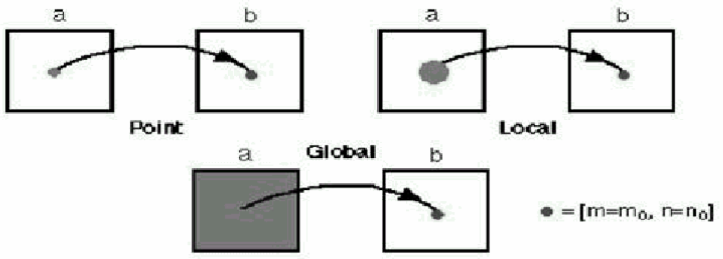


Figure 3.2: Illustration of various types of image operations

Neighborhood operations play a key role in modern digital image processing. It is therefore important to understand how images can be sampled and how that relates to the various neighborhoods that can be used to process an image.

Rectangular sampling - In most cases, images are sampled by laying a rectangular grid over an image as illustrated in Figure 3.1. This results in the type of sampling shown in Figure 3.3(a,b).

Hexagonal sampling - An alternative sampling scheme is shown in Figure 3.3c and is termed hexagonal sampling.

Both sampling schemes have been studied extensively and both represent a possible periodic tiling of the continuous image space. We will restrict our attention, however, to only rectangular sampling as it remains, due to hardware and software considerations, the method of choice. Local operations produce an output pixel value b[m=mo,n=no] based upon the pixel values in the neighborhood of a[m=mo,n=no]. Some of the most common neighborhoods are the 4-connected neighborhood and the 8-connected neighborhood in the case of rectangular sampling and the 6-connected neighborhood in the case of hexagonal sampling illustrated in Figure 3.3.

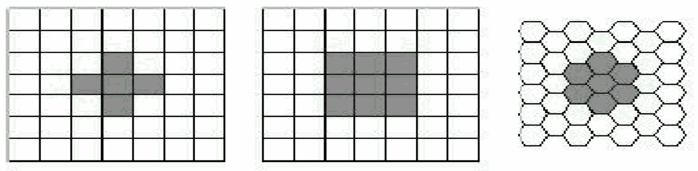


Figure 3.3: Some common neighborhoods: (a) Rectangular sampling, (b) Rectangular sampling, (c) Hexagonal sampling, 4-connected, 8-connected, 6-connected

3.3 Pre-Processing Techniques

The input of the system, which is output of a camera, is considered as raw data. Raw data itself is not sufficient to extract the useful information. Therefore, some pre-processing methods must be applied to the raw input image.

3.3.1 Color spaces & color space conversion

Natural color images, as opposed to computer-generated images, usually originate from a color scanner or a color video camera. These devices incorporate three sensors that are spectrally sensitive to the red, green, and blue portions of the light spectrum. The color sensors typically generate red, green, and blue color signals that are linearly proportional to the amount of red, green, and blue light detected by each sensor. Linear RGB images are the basis for the generation of the various color space image representations. Figure 3.4 shows the “space” defined by RGB signals: it is a Cartesian cubic space, since the red, green, and blue signals are independent and can be added to produce any color within the cube. There are other encoding schemes that are more useful for image processing, since they are more closely related to human perception.

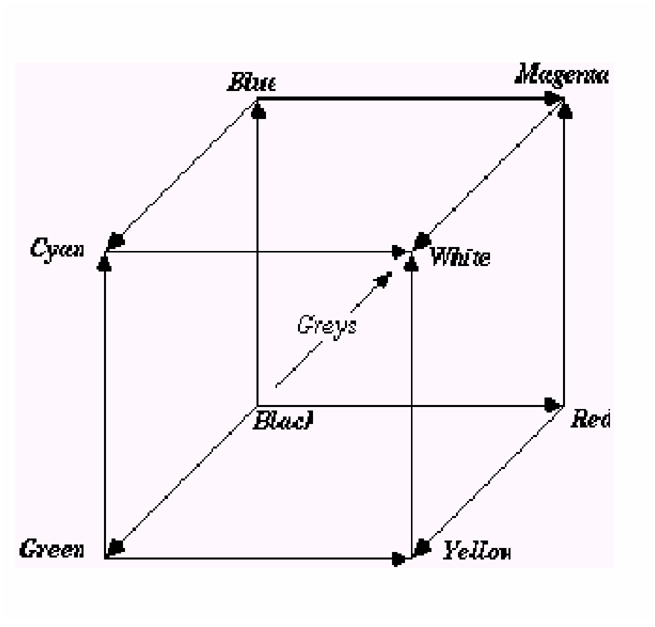


Figure 3.4: RGB Color Space

Raw data generally is in the form of RGB since this model is good for color representation on a color monitor. However, RGB representation is not an appropriate color space to extract the color and gray information for an image. So, color space conversion is required.

Conversion from RGB (the brightness of the individual red, green, and blue signals at defined wavelengths) to YIQ/YUV and to the other color encoding schemes is straightforward and loses no information. Y, the “luminance” signal, is just the brightness of a panchromatic monochrome image that would be displayed by a black-and-white television receiver. It combines the red, green, and blue signals in proportion to the human eye’s sensitivity to them. The I and Q (U and V) components of the color signal are chosen for compatibility with the hardware used in broadcasting; the I signal is essentially red minus cyan, while Q is magenta minus green. The relationship between YIQ and RGB is shown in Table 3.3. An inverse conversion from the encoded YIQ signal to RGB simply requires inverting the matrix of values.

Table 3.3: Conversions of RGB and YIQ color scales or vice versa

Y = 0.299 R + 0.587 G + 0.114 B	R = 1.000 Y + 0.956 I + 0.621 Q
I = 0.596 R – 0.274 G – 0.322 B	G = 1.000 Y – 0.272 I – 0.647 Q
Q = 0.211 R – 0.523 G + 0.312 B	B = 1.000 Y – 1.106 I + 1.703 Q

There are some other representations, which separate the color and gray scale information and represent them as different components. HSL (Hue, Saturation, Lightness), HSI (Hue, Saturation, Intensity) and HSV (Hue, Saturation, Value) are most widely used representations. These are closely related to each other. In this system, hue is the color as described by wavelength, for instance the distinction between red and yellow. Saturation is the amount of the color that is present, for instance the distinction between red and pink. The third axis (called lightness, intensity or value) is the amount of light, the distinction between a dark red and light red or between dark grey and light grey. The only difference between these models is the measurement of saturation, or the strength of the color.

The conversion from RGB to HSI is as follows:

$$I=\frac{(R+G+B)}{3} \tag{3.1}$$

$$H=\cos^{-1}\left(\frac{\left(\frac{(R-G)}{2}+(R-B)\right)}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right) \tag{3.2}$$

$$S=1-a\times\left(\frac{3}{(R+G+B)}\right) \tag{3.3}$$

where a is the minimum of R,G and B.

3.3.2 Thresholding

Selecting features within a scene or image is an important prerequisite for most kinds of measurement or understanding of the scene. Traditionally, one simple way thresholding has been accomplished is to define a range of brightness values in the original image, select the pixels within this range as belonging to the foreground, and reject all of the other pixels to the background. Such an image is then usually displayed as a binary or two-level image, using black and white colors. This operation is called thresholding which is also called as binarization.

This technique is based upon a simple concept. A parameter "T" called the brightness threshold is chosen and applied to the image a [m, n] as follows:

If a [m, n] >= T a [m, n] = object =1

Else a [m, n] = background = 0 (3.4)

This version of the algorithm assumes that we are interested in light objects on a dark background. For dark objects on a light background we would use:

If a [m, n] < T a [m, n] = object =1

Else a [m, n] = background = 0 (3.5)

The main question in thresholding then becomes: How do we choose the threshold value? While there is no universal procedure for threshold selection that is guaranteed to work on all images, there are a variety of alternatives.

Fixed threshold: One alternative is to use a threshold that is chosen independently of the image data. If it is known that one is dealing with very high-contrast images where the objects are very dark and the background is homogeneous and very light, then a constant threshold of 128 on a scale of 0 to 255 might be sufficiently accurate.

Histogram-derived thresholds: In most cases the threshold is chosen from the brightness histogram of the region or image that we wish to segment.

3.4 Image Enhancement

Image enhancement processes consist of a collection of techniques that seek to improve the visual appearance of an image or to convert the image to a form better suited for analysis by a human or a machine.

There is no general unifying theory of image enhancement at present because there is no general standard of image quality that can serve as a design criterion for an image enhancement processor. Consideration is given here to a variety of techniques that have proved useful for human observation improvement and image analysis.

3.4.1 Histogram-based operations

The histogram of an image is simply a graph of brightness values versus number of pixels having that value. This is an important tool for image analysis giving the general information about the image.

3.4.1.1 Contrast stretching

Frequently, an image is scanned in such a way that the resulting brightness values do not make full use of the available dynamic range. By stretching the histogram over the available dynamic range we attempt to correct this situation. If the image is intended to go from brightness 0 to brightness $2^B - 1$, then one generally maps the 0% value to the value 0 and the 100% value (or maximum) to the value $2^B - 1$. The appropriate transformation is given by:

$$b[m,n] = (2^B - 1) \times \frac{a[m,n] - \text{minimum}}{\text{maximum} - \text{minimum}} \tag{3.6}$$

3.4.1.2 Histogram equalization

The intensity values of a typical image are often distributed un-evenly across the full range of 0 to 255 (for an 8-bit image), with most the mass near mid-gray (128) and falling of on either side. An image can be transformed so that the distribution of intensity values is at, that is, each intensity value is equally represented in the image. This process is known has histogram equalization.

This algorithm makes the distribution that large range is assigned for the intensities having more number of pixels and narrow range is assigned for the intensities having less number of pixels.

3.4.2 Image smoothing

Smoothing operations are used primarily for diminishing spurious effects that may be present in a digital image as a result of a poor sampling system or transmission channel.

3.4.2.1 Neighborhood averaging

Neighborhood averaging is a straightforward spatial-domain technique for image smoothing. Given an N x N image *a(x, y)*, the procedure is to generate a smoothed image *b(x, y)* whose gray level at every point *(x, y)* is obtained by averaging the gray-level values of the pixels of *a* contained in a predefined neighborhood of *(x, y)*. In other words, the smoothed image is obtained by using the relation:

$$b(x,y)=\frac{1}{M}\sum_{(n,m)\in S} a(n,m) \tag{3.7}$$

for x, y = 0,1,….N-1. S is the set of coordinates of points in the neighborhood of the point (x, y), including (x, y) itself, and M is the total number of points in the neighborhood.

3.4.2.2 Ordering-based filtering

One of the principal disadvantages of neighborhood algorithm is that it blurs edges and other sharp details. To overcome this disadvantage, median filtering is used as an alternative approach.

Median filters are filters in which we replace the gray level of each pixel by the median of the gray levels in a neighborhood of that pixel, instead of by the average. This method is particularly effective when the noise pattern consists of strong components, and where the characteristic to be preserved is edge sharpness. In order to perform median filtering in a neighborhood of a pixel we sort the values of the pixel and its neighbors, determine the median, and assign this value to the pixel. For example, in a 3 x 3 neighborhood, the median is the 5th largest value, in a 5 x 5 neighborhood the 13th largest value, and so on.

The other order-based filters are minimum filters and maximum filters. In minimum filtering, in a neighborhood of a pixel the values of the pixels are ordered, then a pixel value that has minimum gray level is assigned to the pixels in the neighborhood. This algorithm increases the dark regions of the image in size and decreases the more bright regions in size. On the other hand, maximum filtering algorithm is a method that in e neighborhood the values of the pixels are ordered from maximum value to minimum, then maximum-valued gray level is assigned to the pixels in the neighborhood. This filtering makes the opposite effect of minimum filtering.

3.4.2.3 Low-pass filtering

In digital images, the term “image frequency” refers to the changes of gray levels of pixels in an image plane. In other words, frequency is a measure of change in brightness from a pixel to the successive neighbor pixel.

To eliminate the noises in an image (high-frequency content of the image), low-pass filters are used. This filtering algorithm is made using masks (also referred windows or filters). Basically a mask is a small (e.g. 3 x 3) two-dimensional array, as shown in Figure 3.5, whose coefficients are chosen to detect a given property in an image.

w1	w2	w3
w4	w5	w6
w7	w8	w9

Figure 3.5: A sample mask with 3 x 3 size.

The procedure of masking is as follows: The center of the mask is moved around the image. At each pixel position in the image, every pixel that is contained within the mask area is multiplied by the corresponding mask coefficient. The results of these multiplications are then summed giving the new value of the pixel.

If we let w1, w2, w3,…., w9 represent mask coefficients and consider the 8-neighbors of (x, y), we may generalize the operation as:

$$\begin{aligned} T[\,a(x, y)\,] = {} & w1 * a(x-1, y-1) + w2 * a(x-1, y) + w3 * a(x-1, y+1) \\ & + w4 * a(x, y-1) + w5 * a(x, y) + w6 * a(x, y+1) + \\ & + w7 * a(x+1, y-1) + w8 * a(x+1, y) + w9 * a(x+1, y+1) \qquad (3.8) \end{aligned}$$

on a 3 x 3 neighborhood of (x, y), as shown in Figure 3.6.

w1 (x-1,y-1)	W2 (x-1,y)	w3 (x-1,y+1)
w4 (x,y-1)	W5 (x, y)	w6 (x,y+1)
w1 (x+1,y-1)	W8 (x+1,y)	w9 (x+1,y+1)

Figure 3.6: A pictorial representation masking, a 3 x 3 mask showing coefficients and corresponding image pixel locations.

Larger masks are formed in a similar manner.

In order to attenuate a specified range of high-frequency components in an image, low-pass filter can be applied as the mask shown in Figure 3.7.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Figure 3.7: A 3 x 3 Low-pass Filter coefficients.

If this filter is applied to a region where it has big changes in gray-level values in an image, the differences will be decreased between pixels in that region. Because, this filtering algorithm assigns the average value of neighbors gray-level values to the center pixel that has been intended to smooth. However, this filter makes no chance in the region where there is no difference between the pixel gray-levels.

This filtering algorithm is effective for eliminating the noises in an image, but it blurs the image as an adverse effect.

3.4.2.4 High-pass filtering

If an application desires to point out the regions where there are big gray-level differences in the pixels of the region, then low-frequency content of the image must be removed, only high-frequency-valued regions must remain. This is achieved using high-pass filtering.

For doing this, in the mask (filter), greater-valued coefficient in the centre and negative-valued coefficients in the surroundings is located as shown in Figure 3.8.

-1	-1	-1
-1	9	-1
-1	-1	-1

Figure 3.8: A 3 x 3 High-pass Filter coefficients.

If there is no difference in pixel gray-level values in the region, there will be no change in the pixels. However, if this filter is applied to the region where it has big changes in the pixel gray-level values, then the region having these changes becomes more apparent and more evident.

3.4.3 Morphological operations

Morphological operations are methods for processing binary images based on shapes. These operations take a binary image as input, and return a binary image as output. The value of each pixel in the output image is based on the corresponding input pixel and its neighbors. By choosing the neighborhood shape appropriately, you can construct a morphological operation that is sensitive to specific shapes in the input image.

Each morphological operation uses a specified neighborhood. The state of any given pixel in the output image is determined by applying a rule to the neighborhood of the corresponding pixel in the input image. The neighborhood is represented by a structuring element, which is a matrix consisting of only 0's and 1's. The center pixel in the structuring element represents the pixel of interest, while the elements in the matrix that are on (i.e., = 1) define the neighborhood.

There are some morphological operations including dilation, erosion, opening and closure operations.

3.4.3.1 Dilation

Dilation is an operation which adds pixels to the boundaries of objects (i.e., changes them from off to on). In other words, if any pixel in the input pixel's neighborhood is on, the output pixel is on. Otherwise, the output pixel is off.

Dilation operation can be formulized as:

$$D(A,B)=A\oplus B=\bigcup_{\beta\in B}(A+\beta) \tag{3.9}$$

While either set A or B can be thought of as an "image", A is usually considered as the image and B is called a structuring element. Dilation operation can be explained schematically shown in Figure 3.9.

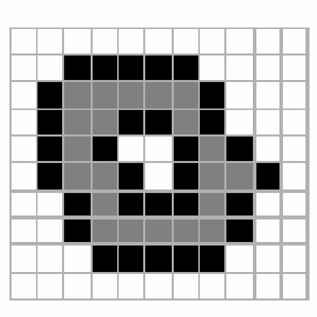


Figure 3.9: Illustration of dilation operation

In this figure, the dilation operation is shown as: Original object pixels are in gray; pixels added through dilation are in black.

As a conclusion, dilation operation causes objects to dilate or grow in size.

3.4.3.2 Erosion

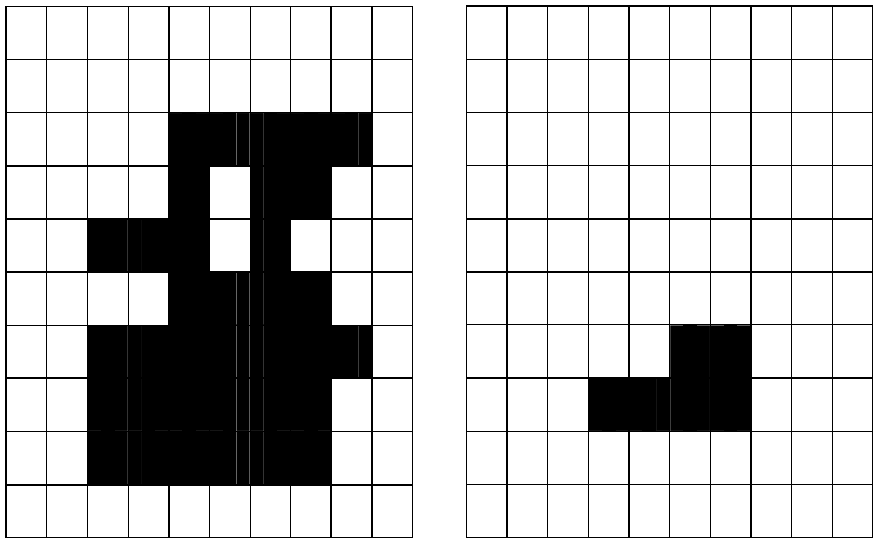
In contrast with the dilation operation, erosion operation removes pixels on object boundaries (changes them from on to off). That is to say, for erosion, if every pixel in the input pixel's neighborhood is on, the output pixel is on. Otherwise, the output pixel is off.

Erosion operation applies the mathematical operation as:

$$E(A,B)=A\Theta(-B)=\bigcap_{\beta\in B}(A-\beta) \qquad (3.10)$$

where $-B=\{-\beta|\beta\in B\}$

Erosion operation can be represented as in Figure 3.10.



(a) (b)

Figure 3.10: Pictorial representation of erosion operation, (a)- Original Image (b)- Output Image after erosion operation.

It can be said that as a conclusion, erosion operation causes objects to shrink.

3.4.3.3 Opening and closing operations

As it has been seen that, dilation operation expands the image and erosion operation shrinks the image. There are some other important operations which are just modified version of dilations or erosions, or combinations of dilation and erosion as mainly opening and closing operation.

Opening operation consists of erosion followed by dilation operation, symbolically as;

$$O(A,B)=A\circ B=(A\Theta B)\oplus B \tag{3.11}$$

where B is a structuring element.

Opening of an image generally smoothes the contour of an object, breaks narrow strokes and eliminates thin protrusions.

As opposed to opening operation, dilation followed by erosion operation is called a closure (closing) operation. It is expressed symbolically as;

$$C(A,B)=A\bullet B=(A\oplus B)\Theta B \tag{3.12}$$

where B is a structuring element.

Closing of an image with a compact structuring element also smoothes section of contours of objects but as opposed to opening, it eliminates small holes in the objects, and fuses short gaps between objects.

When dialing with binary images, the principal application of morphological operations is extracting image components that are useful in the representation and description of the shapes in the images. Some of the algorithms in morphology are the algorithms for extracting boundaries, region filling, extraction of connected components, thinning, thickening, finding skeleton of a region, pruning, etc.

3.5 Segmentation

One of the most widely used steps in the process of reducing images to information is segmentation: dividing the image into regions that hopefully correspond to structural units in the scene or distinguish objects of interest. Segmentation is often described by analogy to visual processes as a foreground/background separation, implying that the selection procedure concentrates on a single kind of feature and discards the rest.

Segmentation subdivides an image into its constituent regions or objects. The level to which the subdivision is carried depends on the desired interest. In other words, segmentation algorithm should be terminated when the objects of interest in an application have been isolated.

Image segmentation algorithms are generally based on one of the two basic properties of intensity values: discontinuity and similarity. In the first case, the approach is to partition an image based on abrupt changes in intensity, such as edges in an image. But in the second case, the basic approaches are based on partitioning an image into regions that are similar according to a set of predefined criteria, as in the thresholding or smearing algorithms.

3.5.1 Finding discontinuities

Considering the digital images, it has been seen that there are three basic types of gray-level discontinuities as the points, lines and the edges. Also, there has been several ways to detect the discontinuities in a digital image. But the most common way to detect the discontinuities is to use masking algorithm. Masking algorithm involves the use of appropriate mask for the desired type of discontinuities that is wanted and to run the mask through the target region of the image. Masking algorithm has been defined and explained in filtering algorithms sections of the thesis.

3.5.1.1 Detection of points

The simplest and the most effective way to detect the isolated points in an image is to use a mask that is useful for finding the point or points whose gray level are obviously different from its background or surroundings. Therefore, the mask used in high-pass filtering can easily detect such an isolated point or points located in homogeneous or nearly homogeneous region. An example of 3x3 mask for the purpose is shown in Figure 3.11.

-1	-1	-1
-1	8	-1
-1	-1	-1

Figure 3.11: A 3x3 mask for point detection.

On the other hand, this mask is different from the high-pass filtering case that it makes the points in the surroundings that is homogeneous (whose gray-level values are equal) to zero indicating the isolated point or points. This is the change of centered coefficient of the mask to 8 instead of 9 for 3x3 masking.

3.5.1.2 Detection of lines

The line in an image can be distinguished from the surroundings with having different gray-level values. The line can be horizontal, vertical or in any orientation. If orientation of the line to be considered is known, then the mask that will respond the more strongly to the line pixels is used. The coefficients of the mask is selected that the line pixels are more evident (bigger coefficients) and the others(no line) are zero. If the line is horizontal, the mask shown in Figure 3.12 can demonstrate the line.

-1	-1	-1
2	2	2
-1	-1	-1

Figure 3.12: A sample mask for horizontal lines

Here if the mask is applied through the image having horizontal line with a constant background, the maximum response will be obtained when the line passed through the middle row of the mask.

The similar approach can be applied for finding vertical lines, the lines oriented at +45 degrees or the lines oriented at -45 degrees with the masks given in Figure 3.13 respectively.

-1	2	-1
-1	2	-1
-1	2	-1

Vertical

-1	-1	2
-1	2	-1
2	-1	-1

Oriented at +45º

2	-1	-1
-1	2	-1
-1	-1	2

Oriented at -45º

Figure 3.13: Line Masks for Various Orientations

If the all possible lines that may be oriented horizontally, vertically, or in any direction are wanted to detect, then all possible masks are applied through the image successively. Then the thresholding is made through the image and the pixels that are in the lines are pointed out as the detection.

3.5.1.3 Detection of edges

The edge in an image is the sudden change from dark to light or from light to dark pixel intensity. In other words, edges are the places in an image that correspond the object boundaries. To find those pixels that belong to the borders of the objects is called as edge detection.

To find edges looking for places in the image where the intensity changes rapidly uses one of these two criteria:

- Places where the first derivative of the intensity is larger in magnitude than some threshold.
- Places where the second derivative of the intensity has a zero crossing.

We use these operators because of the fact that the magnitude of first derivative can be used to detect the presence of an edge at a point in an image (i.e., to determine if a point is on a ramp). Similarly, the sign of second derivative can be used to determine whether an edge pixel lies on the dark or light side of an edge with properties of second derivative around an edge that an imaginary straight line joining the extreme positive and negative values of the second derivative would cross zero near the midpoint of the edge. This zero-crossing property of the second derivative is quite useful for locating the centers of thick edges.

First-order derivatives in an image can be computed using the gradient operators and second-derivatives are obtained using the Laplacian. The gradient of an image f(x,y) at location (x,y) is defined as:

$$\nabla f = [G_x ; G_y] = \left[\frac{\partial f}{\partial x} ; \frac{\partial f}{\partial y}\right] \tag{3.13}$$

and the magnitude of the gradient denoted by,

$$\mathrm{mag}(\nabla f) = \left[G_x^2 + G_y^2\right]^{1/2} \tag{3.14}$$

Computation of the gradient of an image is based on obtaining the partial derivatives $\partial f/\partial x$ and $\partial f/\partial y$ at every pixel location.

If a 3x3 area represents the gray levels in a neighborhood of an image shown in figure 3.14 (a), one of the ways to calculate the first-order partial derivative at point z5 is to use the Roberts cross-gradient operators:

$$G_x = (z9 - z5) \quad \text{and} \quad G_y = (z8 - z6) \qquad (3.15)$$

with the mask given in Figure 3.14 (b).

Z1	z2	Z3
Z4	z5	Z6
Z7	z8	Z9

-1	0
0	1

0	-1
1	0

(a) (b)

Figure 3.14: (a) A 3x3 region of an image , (b)Roberts mask

And similar approach using the 3x3 mask in Figure 3.15 can be used to give the partial derivatives as:

$$\begin{aligned} G_x &= (z7+z8+z9) - (z1+z2+z3) \\ G_y &= (z3+z6+z9) - (z1+z4+z7) \end{aligned} \qquad (3.16)$$

Here the difference between the first and third rows of the 3x3 image region approximates the derivative in x-direction, and the difference between the third and first columns approximates the derivative in y-direction.

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Figure 3.15: Prewitt masks for edge detection

Some small changes in the mask coefficients to achieve smoothing by giving more importance to the center point called Sobel masks is shown in Figure 3.16.

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Figure 3.16: Sobel masks for edge detection

The Sobel mask provides superior noise-suppression characteristic than the Prewitt masks.

The Laplacian of a an image is a second-order derivative defined as:

$$\nabla f^{2} = \left[\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right] \qquad (3.17)$$

The Laplacian is not used in its original form for edge detection because of being sensitive to noise. So, the Laplacian can be combined with smoothing as a precursor to find edges via zero-crossings. Here is the example of 5x5 Laplacian of Gaussian mask in Figure 3.17.

0	0	1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

Figure 3.17 : A 5x5 mask for edge detection using Laplacian of Gaussian function.

As a summary, the Sobel method finds edges using the Sobel approximation to the derivative. It returns edges at those points where the gradient of an image is maximum. The Prewitt method finds edges using the Prewitt approximation to the derivative. It returns edges at those points where the gradient of an image is maximum. The Roberts method finds edges using the Roberts approximation to the derivative. It returns edges at those points where the gradient of an image is maximum. The Laplacian of Gaussian method finds edges by looking for zero crossings after filtering an image with a Laplacian of Gaussian filter. The zero-cross method finds edges by looking for zero crossings after filtering an image with a filter you specify [41].

3.5.2 Smearing algorithm

Smearing is another type of segmentation method for the extraction of text areas on a mixed image. This algorithm is also called as Run-Length Smearing (RLS) or Run-Length Smoothing Algorithm (RLSA).

With the smearing algorithm, the image is processed along vertical and horizontal runs (scan-lines). If an image consists of only 1's and 0's (only black and white) called as binary image and if the number of white pixels is less than a desired threshold or greater than any other desired threshold, white pixels are converted to black or vice versa. The smearing algorithm can be made horizontally or vertically through the image.

The white pixels (gaps) between the letters or words in the case of text areas and the white pixels within the letters are converted to black if the pixel/pixel groups are less or greater than the threshold or vice versa (converting the black to white).

This algorithm can be formulated as:

If number of 'white'/ 'black' pixels < threshold ; pixels become 'black' / 'white'
Else ; no change
OR
If number of 'white' / 'black' pixels > threshold ; pixels become 'black' / 'white'
Else ; no change (3.18)

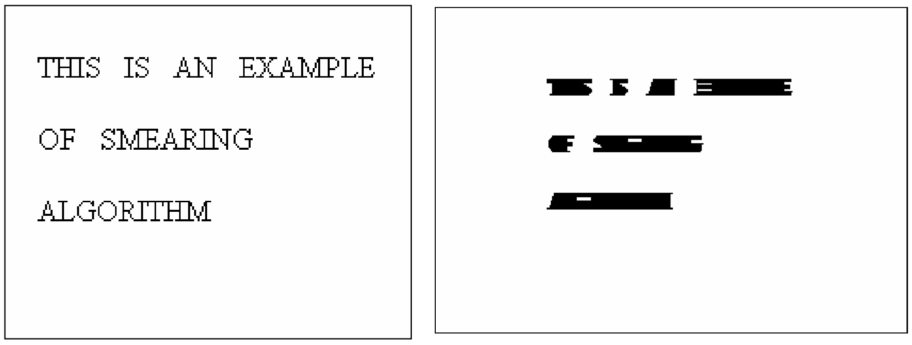
For example, if the pixel values for a horizontal region is given as:

11000111110111111100000111000000000111100001 0011111

and if the threshold value is chosen as 5 and the desired operation is to smear the white pixel / pixels smaller than the threshold (changing the white pixels (1’s) to black (0’s)), then the sequence becomes after the smearing as:

000001111101111111000000000000000000000000000011111

And, an example of horizontal smearing algorithm that is used for detecting word regions by selecting the appropriate threshold value is shown in Figure 3.18. In this example the selected threshold value is 10 meaning that if the number of white pixels is less than the threshold value,10, the pixels become black.



(a) (b)

Figure 3.18 : (a) Original image containing text , (b) The output image after smearing

3.5.3 Segment labeling

The result of any successful image segmentation is the labeling of each pixel that lies within a specific distinct segment. One means of labeling is to append to each pixel of an image the label number or index of its segment. A more succinct method is to specify the closed contour of each segment. If necessary, contour filling techniques can be used to label each pixel within a contour. The following describes the common technique of contour following.

In the following binary image of Figure 3.19 as an example, a conceptual contour follower begins marching from the white background to the black pixel region indicated by the closed contour. When the contour follower crosses into a black pixel, it makes a left turn and proceeds to the next pixel. If that pixel is black, the follower again turns left, and if the pixel is white, the follower turns right. The procedure continues until the follower returns to the starting point. This simple contour follower may miss spur pixels on a boundary.

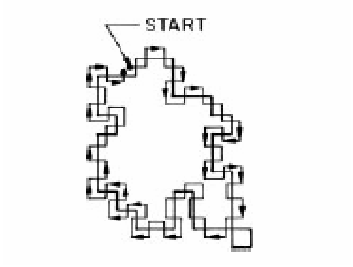


Figure 3.19 : Contour Following

The other technique is known as backtracking contour follower. In this algorithm, if the follower makes a white-to-black pixel transition, it returns to its previous starting point and makes a right turn. The follower makes a right turn whenever it makes a white-to-white transition.

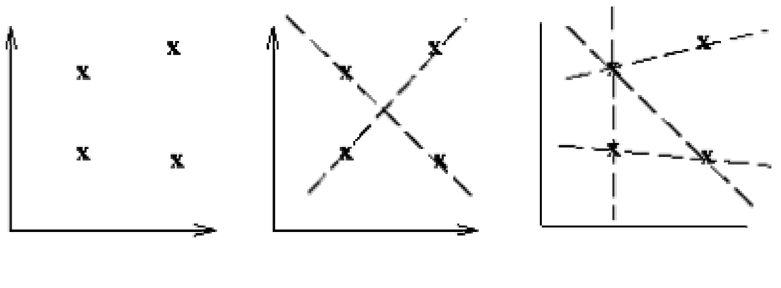
While the contour follower is following a contour, it can create a list of the pixel coordinates of each boundary pixel. Alternatively, the coordinates of some reference pixel on the boundary can be recorded, and the boundary can be described by a relative movement code. That is pixels that have been dealt with are labeled using some coding techniques [42].

3.5.4 Hough transform

The Hough transform is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the classical Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, etc.

To understand the Hough Transform, it is important to know what the Hough space is. Each point (r,θ) in Hough space corresponds to a line at angle θ and distance r from the origin in the original data space. The value of a function in Hough space gives the point density along a line in the data space. The Hough Transform utilizes this by the following method.

For each point in the original space consider all the lines which go through that point at a particular discrete set of angles, chosen a priori. For each angle θ, calculate the distance to the line through the point at that angle and discretize that distance using an a priori chosen discretization, giving value r. Make a corresponding discretization of the Hough space - this will result in a set of boxes in Hough space. These boxes are called the Hough accumulators. For each line we consider above, we increment a count (initialised at zero) in the Hough accumulator at point (r,θ). After considering all the lines through all the points, a Hough accumulator with a high value will probably correspond to a line of points. In fact for uniformly distributed points, each Hough box should have a Poisson distributed count with mean given by the length of the line times discretisation width times uniform point density. A count which is in the tail of the relevant Poisson distribution is unlikely to be the result of the underlying uniform data, and hence more likely to be the result of some line of points. Giving some prior model for the number of points in a line will allow a proper Bayesian assessment of whether there is a line at the relevant angle and distance from the origin. As a simple example, considering the common problem of fitting a set of line segments to a set of discrete image points (e.g. pixel locations output from an edge detector). Figure 3.20 shows some possible solutions to this problem. Here the lack of a priori knowledge about the number of desired line segments (and the ambiguity about what constitutes a line segment) render this problem under-constrained.



(a) (b) (c)

Figure 3.20: (a) Coordinate points, (b) and (c) Possible straight line fittings.

We can analytically describe a line segment in a number of forms. However, a convenient equation for describing a set of lines uses parametric or normal notion:

$$x\cos\theta + y\sin\theta = r \tag{3.19}$$

where r is the length of a normal from the origin to this line and θ is the orientation of r with respect to the x-axis as in Figure 3.21. For any point (x, y) on this line r and θ are constant.

.

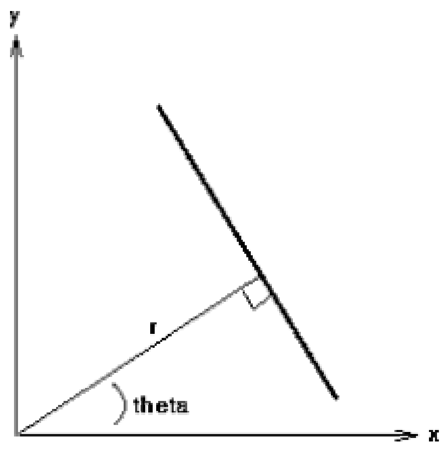


Figure 3.21: Parametric description of a straight line

In an image analysis context, the coordinates of the point(s) of edge segments (i.e. (x_i, y_i)) in the image are known and therefore serve as constants in the parametric line equation, while r and θ are the unknown variables we seek. If we plot the possible (r,θ) values defined by each (x_i, y_i), points in cartesian image space map to curves (i.e. sinusoids) in the polar Hough parameter space. This point-to-curve transformation is the Hough transformation for straight lines. When viewed in Hough parameter space, points which are collinear in the cartesian image space become readily apparent as they yield curves which intersect at a common (r,θ) point.

The transform is implemented by quantizing the Hough parameter space into finite intervals or accumulator cells. As the algorithm runs, each (x_i, y_i) is transformed into a discretized (r,θ) curve and the accumulator cells which lie along this curve are incremented. Resulting peaks in the accumulator array represent strong evidence that a corresponding straight line exists in the image [43].

3.6 Character Recognition

Optical character recognition, usually abbreviated to OCR, involves computer software designed to translate images of typewritten text (usually captured by a scanner) into machine-editable text, or to translate pictures of characters into a standard encoding scheme representing them in (ASCII or Unicode). OCR began as a field of research in artificial intelligence and machine vision. Though academic research in the field continues, the focus on OCR has shifted to implementation of proven techniques.

Human performs recognition operation as they read, taking the printed character, interpreting its shape, and assigning meaning. On the other hand, computers cannot normally achieve this intelligence and instead they must be taught how to recover text from a character image.

In 1950, David Shepard, a cryptanalyst at AFSA, the forerunner of the United States National Security Agency (NSA), was asked by Frank Rowlett, who had broken the Japanese PURPLE diplomatic code, to work with Dr. Louis Tordella to recommend data automation procedures for the Agency. This included the problem of converting printed messages into machine language for computer processing. Shepard decided it must be possible to build a machine to do this, and, with the help of Harvey Cook, a friend, built "Gismo" in his attic during evenings and weekends. Shepard then founded Intelligent Machines Research Corporation (IMR), which went on to deliver the world's first several OCR systems used in commercial operation. While both Gismo and the later IMR systems used image analysis, as opposed to character matching, and could accept some font variation, Gismo was limited to reasonably close vertical registration, whereas the following commercial IMR scanners analyzed characters anywhere in the scanned field, a practical necessity on real world documents. The first commercial system was installed at the Readers Digest in 1955, which, many years later, was donated by Readers Digest to the Smithsonian, where it was put on display. The second system was sold to the Standard Oil Company of California for reading credit card imprints for billing purposes, with many more systems sold to other oil companies. Other systems sold by IMR during the late 1950's were a bill stub reader to the Ohio Bell Telephone Company and a page scanner to the U.S. Air Force for reading and transmitting by teletype typewritten messages. IBM and others were later licensed on Shepard's OCR patents [44]. These are the brief history of OCR systems

There are two methods for performing the character recognition: template-based recognition and feature-based recognition. Although there have been hundreds of different algorithm for each of these methods, the principle behind the particular algorithm falls into one of these two categories.

Template-based recognition matches each input character image against a library of known character images called templates. With comparing the input image with the templates, the best similarity gives the correct recognition. But, in feature-based recognition, this approach attempts to compare the input image’s identification features or label with the identification database. The input is identified as the character whose features best matches that of the input image.

3.6.1 Template matching

One of the most fundamental means of object detection within an image field is by template matching, in which a replica of an object of interest is compared to all unknown objects in the image field. If the template match between an unknown object and the template is sufficiently close, the unknown object is labeled as the template object.

Template matching is an effective algorithm for recognition of characters. The character image is compared with the ones in the database and the best similarity is measured.

To measure the similarity and find the best match, a statistical method correlation is used. Correlation is an effective technique for image recognition which was developed by Horowitz. [45]. This method measures the correlation coefficient between a number of known images with the same size unknown images or parts of an image with the highest correlation coefficient between the images producing the best match.

There are two forms of correlations: auto-correlation and cross-correlation. Auto-correlation function (ACF) involves only one signal and provides information about the structure of the signal or its behavior in the time domain. Cross-correlation function (CCF) is a measure of the similarities or shared properties between two signals. Since there are two signals as unknown input image and known database image in this system, cross-correlation is used.

If we have an image denoted by $F(j,k)$ to be searched and template which is denoted by $T(j,k)$ for $1 \le j \le J$ and $1 \le k \le K$, then the normalized cross-correlation between the image pair is defined as:

$$R(m,n)=\frac{\sum_{j}\sum_{k}F(j,k)T(j-m+(M+1)/2,k-n+(N+1)/2)}{\left[\sum_{j}\sum_{k}\left|F(j,k)\right|^{2}\right]^{1/2}\left[\sum_{j}\sum_{k}\left|T(j-m+(M+1)/2,k-n+(N+1)/2\right|^{2}\right]^{1/2}} \quad (3.20)$$

for m=1,2,….M and n=1,2,….N, where M and N are odd integers.

R(m,n) is known as the cross-correlation coefficient. Its value always lies between -1 and +1, meaning that

- +1 means 100 % correlation in the same sense
- -1 means 100 % correlation in the opposing sense
- A value of 0 signifies zero correlation. This means that the signal pairs are completely independent.

3.6.2 Feature extraction

Feature-based recognition is a feature extraction method that computes numeric or symbolic information from the observations of image with analyzing the image.

An image feature is a distinguishing primitive characteristic or attribute of an image. Some features are natural in the sense that such features are defined by the visual appearance of an image, while other, artificial features result from specific manipulations of an image. Natural features include the luminance of a region of pixels and gray scale textural regions. Image amplitude histograms and spatial frequency spectra are examples of artificial features.

When dealing with the character recognition, this algorithm attempts to work with a subset of the features in a character that a human would typically see for the identification of machine-printed characters. If it were expanded to use more features, it would be made correspondingly slower; with the advent of faster microprocessors this fact is not viewed as a crippling problem.

A feature point is a point of human interest in an image, a place where something happens. It could be an intersection between two lines, or it could be a corner, or it could be just a dot surrounded by space. Such points serve to help define the relationship between different strokes. Two strokes could fully cross each other, come together in for example "Y" or a "T" intersection, form a corner, or avoid each other altogether. People tend to be sensitive to these relationships; the fact that the lines in for example "Z" connect in a certain way is more important than the individual lengths of those lines. These relationships are what should be used for character identification, and the feature points can be exploited for the task.

CHAPTER 4

LICENSE PLATE RECOGNITION SYSTEM

In this thesis, the proposed system consists of three major units:

- Extraction of Plate Region
- Segmentation of Plate Characters
- Recognition of Plate Characters

In this chapter, all three units for LPR system are explained successively in details. Also, the input for the system is given.

4.1 Input of The System

In this work, the images for the input to the system are colored images with the size 1200x1600x3. And, captured images are taken with a digital camera from 4-5 meters away from the car. And, the vehicle images captured by the camera all have Turkish license plate. That is to say, this system is designed for identifying the Turkish license plates. And the test images for the input are taken under various illumination conditions.

4.2 Extraction of Plate Region

Extraction of plate region is to find the exact location of the plate in an image and to label or cut the plate region from the input image. Plate region extraction is the first stage in this algorithm.

Firstly, image captured from the camera is converted to the binary image consisting of only 1's and 0's (only white and black) by thresholding the pixel values of 0 (black) for all pixels in the input image with luminance less than threshold value and 1 (white) for all other pixels. This process is known as binarization. A threshold value is selected for the process and binarization maps the image with the threshold and assigns the 1's (white) for the greater values from the threshold and 0's (black) for the smaller than the threshold giving the binary image as the output.

Selection of the threshold value is important concern. This can be done applying uniform histogram equalization. It means that the histogram of the input image is forced to be distributed uniformly. In this system, we use "im2bw" function in Matlab for binarization process. Im2bw function converts an image to a binary image by thresholding. To do this, it converts the input image to grayscale format and then converts this grayscale image to binary by thresholding.

A sample input image and the processed image after the binarization process are given in Figure 4.1 and Figure 4.2, respectively.

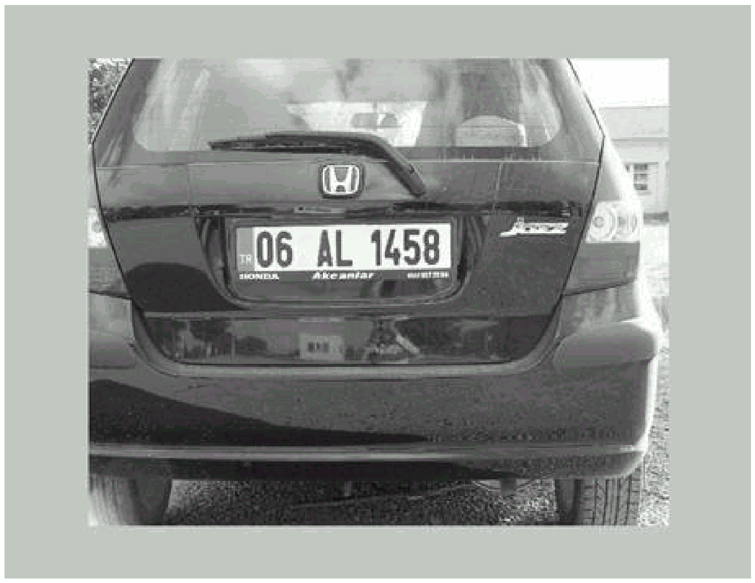


Figure 4.1: A sample image

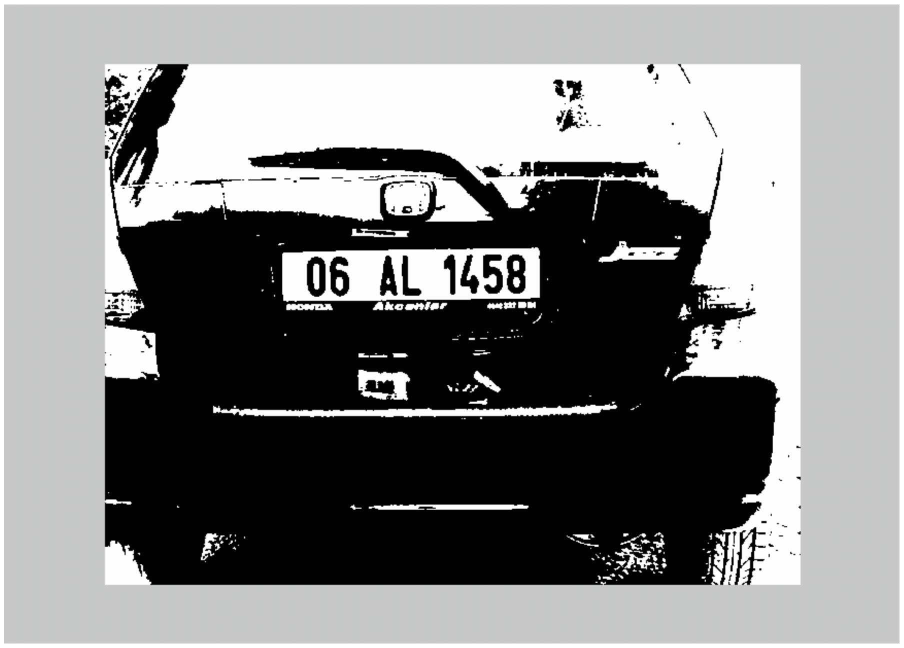


Figure 4.2: Processed image after binarization

After binarization process, the image is filtered with neighborhood averaging algorithm. Neighborhood averaging is a spatial-domain technique for image smoothing. The aim of this algorithm is to generate a smoothed image whose gray level at every point *(x, y)* is obtained by averaging the gray-level values of the pixels of the image contained in a predefined neighborhood of *(x, y)*. This is made using Matlab "fspecial(average)" function and the 5x5 neighborhood is selected. This function creates a 2-D filter and processes the image with averaging algorithm.

Then, to find the plate location, smearing algorithm is used. The plate region is considered as a text region on a mixed image (containing other regions or parts). Therefore, the image is scanned for the plate region (text region) for which the plate region width and height have an upper and lower limit. Since the approximate values of the width and height of the plate are known, an assumption for the limits of the plate region is not a prediction.

Smearing is a method for the extraction of text areas on a mixed image. With the smearing algorithm, the image is processed along vertical and horizontal runs (scan-lines). If the number of white pixels is less than a desired threshold, white pixels are converted to black or if the number of white pixels is greater than a threshold, white pixels are converted to black. In this system, threshold values are selected as 10 and 100 for both horizontal and vertical smearing.

This can be formulized as given below,

If number of ‘white’ pixels < 10 ; pixels become ‘black’
Else ; no change

If number of ‘white’ pixels > 100 ; pixels become ‘black’
Else ; no change (4.1)

The horizontal smearing algorithms written in Matlab are as follows;

```
%horizontal smearing for # of pixels greater than 100
c=0;
t=0;
    for x=1:size(image,1)-1
    c=0;
    t=0;
        for y=1:size(image,2)-1
            if image(x,y)==1
            t=t+1;
               else
               t=0;
               c=0;
            end
                if t>100
                image(x,y)=0;
                      else if t>=100 && c==0
                           for k=y-99:y-1
                           image(x,k)=0;
                          c=1;
                           end
```

```
                    end
                  end
                end
              end


%horizontal smearing for # of pixels smaller than 10
t=0;
c=0;
  for x=1:size(image,1)-1
  t=0;
  c=0;
    for y=1:size(image,2)-1
      if image(x,y)==1
      t=t+1;
        else
        t=0;
        c=0;
      end
        if t<10
        image(x,y)=0;
           else if t>=10 && c==0
             for k=y-9:y-1
             image(x,k)=1;
             c=1;
             end
           end
        end
      end
    end
```

Note that the threshold values are approximated experimentally. After horizontal smearing with the threshold values of 100 and 10, vertical smearing is applied to the image whose codes are as follows;

```
%vertical smearing for # of pixels greater than 100
c=0;
t=0;
    for y=1:size(image,2)-1
    c=0;
    t=0;
        for x=1:size(image,1)-1
           if image(x,y)==1
           t=t+1;
                else
                t=0;
                c=0;
            end
                    if t>100
                    image(x,y)=0;
                         else if t>=100 && c==0
                              for k=x-99:x-1
                              image(k,y)=0;
                              c=1;
                              end
                          end
                     end
                end
            end


%vertical smearing for # of pixels smaller than 10
t=0;
c=0;
    for y=1:size(image,2)-1
```

```
t=0;
c=0;
    for x=1:size(image,1)-1
        if image(x,y)==1
        t=t+1;
            else
            t=0;
            c=0;
        end
            if t<10
            image(x,y)=0;
                    else if t>=10 && c==0
                          for k=x-9:x-1
                          image(k,y)=1;
                          c=1;
                          end
                    end
            end
      end
end
```

After smearing, a morphological operation, dilation, is applied to the image for specifying the plate location. Most probably, these operations find the plate location only. However, there may be sometimes more than one candidate region for plate location. To find the exact region and eliminate the other regions, some criteria tests considering the properties of plate region are applied for the purpose. Then, only plate region remains.

The resulting image containing only plate region is shown in Figure 4.3.

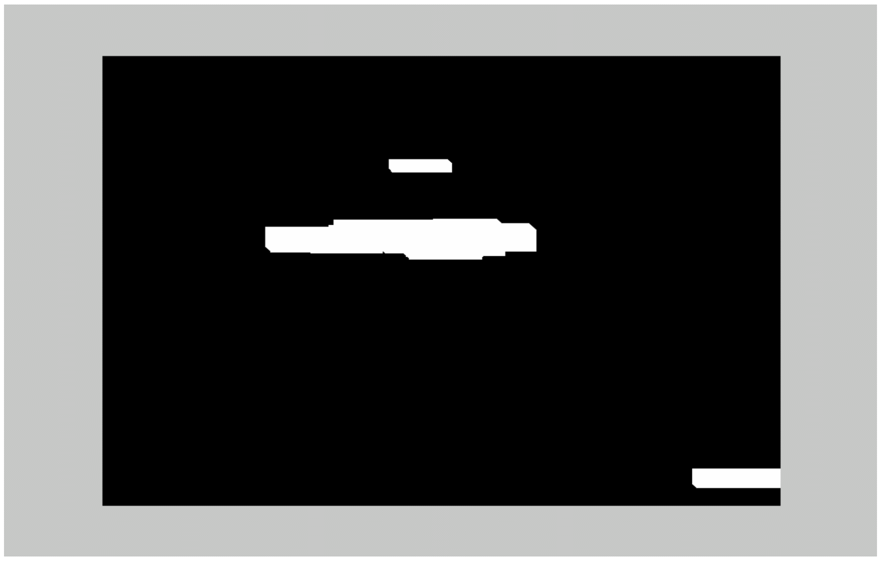


Figure 4.3: Plate Region

Now, the procedure is to obtain an image containing only plate. To do this, lower and upper limits of plate location are found and then, a regular-shaped area is formed with the help of found values of plate boundaries in both vertical and horizontal directions. The resultant image involving only plate region is in Figure 4.4.

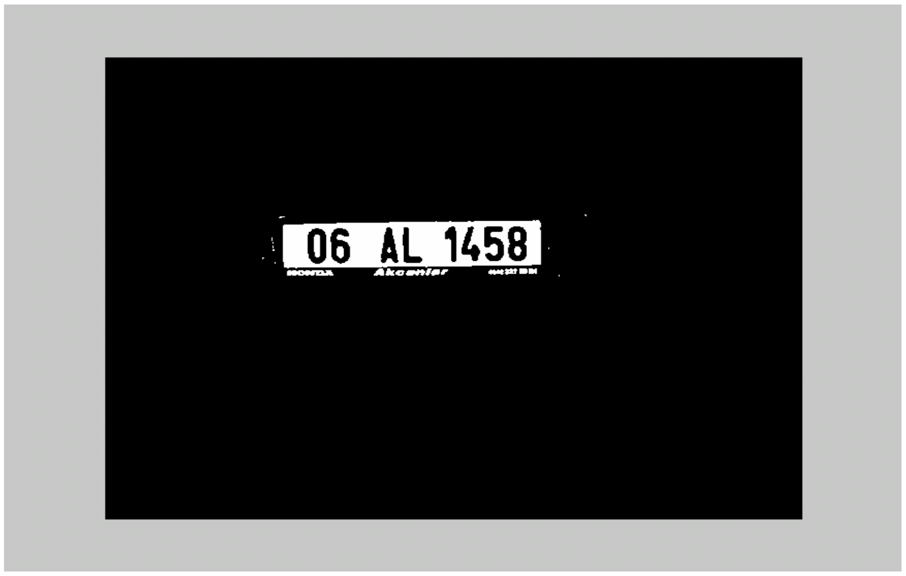


Figure 4.4: Image involving only plate

Some images that have more than one candidate region for plate location can be seen in Figure 4.5.

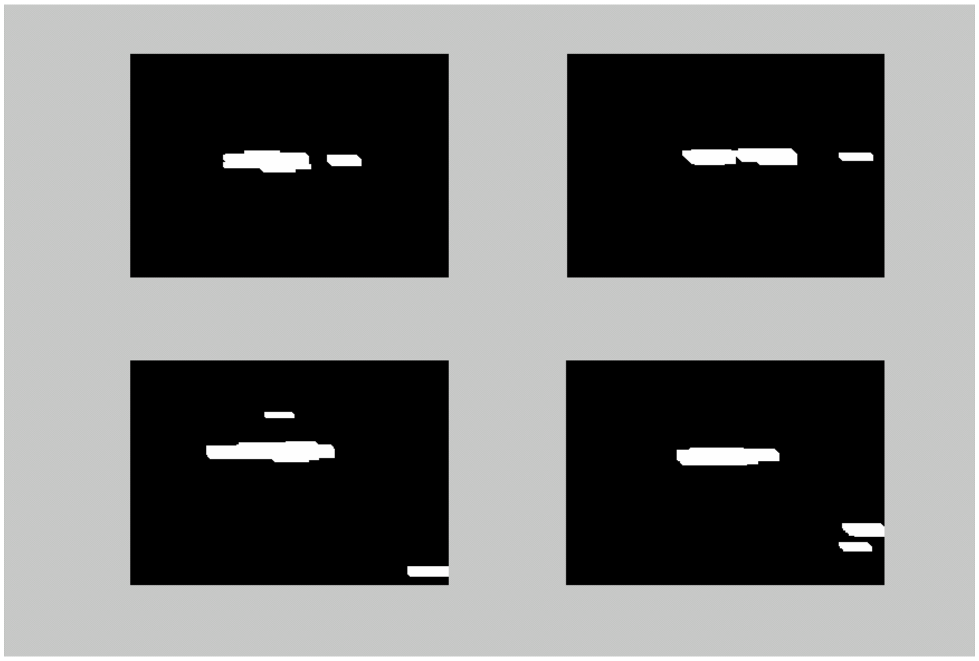


Figure 4.5: Some images having more than one candidate region for plate location.

And, to find the exact plate region, some features of plate region are used. Finally, only one region remains for plate location.

After finding the plate region and obtaining the image involving only plate, the next step is to cut the plate from the original input image. This can be done by calculating the starting and end points of plate in both directions, horizontal and vertical directions. If these values are known, the part that contains only plate is cut. Necessary program segment for finding the starting and end points of plate (plate boundaries) is as follows;

```
% Finding plate boundaries
% In horizontal direction
c=1;
r=0;
for x=1:size(image,1)-1
```

```
    for y=1:size(image,2)-1
        if r~=image(x,y)
        m(c)=y;
        c=c+1;
        r=image(x,y);
        end
     end
     r=0;
end


k=1;
c=1;
for i=1:length(m)/2
t(k)=m(c+1)-m(c);
c=c+2;
k=k+1;
end


% In vertical direction
c=1;
r=0;
for y=1:size(image,2)-1
    for x=1:size(image,1)-1
        if r~=image(x,y)
        n(c)=x;
        c=c+1;
        r=image(x,y);
        end
     end
     r=0;
end
```

```
k=1;
c=1;
for i=1:length(n)/2
s(k)=n(c+1)-n(c);
c=c+2;
k=k+1;
end


[y,i1]=max(t);
[y,i2]=max(s);

% z1,z2,z3 and z4 values are boundaries of plate.
z1=n(2*i2-1);
z2=m(2*i1-1);
z3=n(2*i2);
z4=m(2*i1);
```

In this program segment, the variations of pixels from white to black or black to white are considered, and the maximum values at which the pixels remain white are considered as limit values of plate region. And, by using these boundary values the image is cut to obtain only plate. This cutting operation is made with the help of Matlab "imcrop" function.

Imcrop function crops an image to a specified rectangle. The region that is wanted to cut is specified with the starting end points and these values are given as an input argument, and then the region is obtained.

The resulting image for the sample input is given in Figure 4.6 as an example.

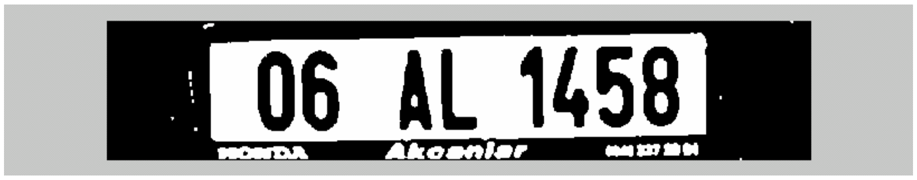


Figure 4.6: Plate image cut from original image

For some other input images, the plates obtained after extraction step can be seen here in Figure 4.7.

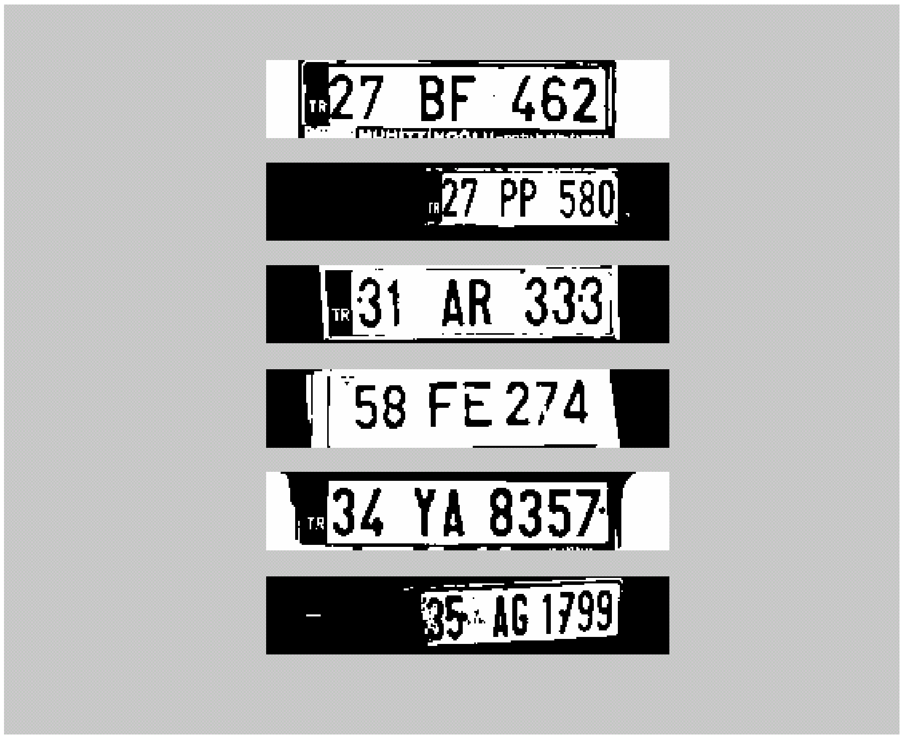


Figure 4.7: Some cutted-plates

4.3 Segmentation of Plate Characters

In the previous step, the place of the license plate is found and this region is cut from the original input image. In this step, it is aimed to extract all of the characters from the license plate.

Namely, in the segmentation of plate characters, license plate is segmented into its constituent parts obtaining the characters individually. Firstly, image is filtered for enhancing the image and removing the noises and unwanted spots using averaging filtering algorithm. Then dilation operation is applied to the image for separating the characters from each other if the characters are close to each other. This is necessary operation. Because, some plate characters can be written in such a way that these characters are very close to each other. To separate these characters, dilation operation is an effective solution.

After filtering and dilation operation, plate image can be seen in Figure 4.8, below.

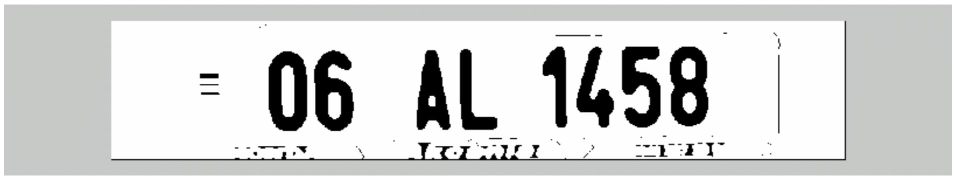


Figure 4.8: Plate image after filtering and dilation operation

The next procedure is to process the image such that characters can be separated easily. To do this, both horizontal and vertical smearing algorithms are applied for finding the character regions. Firstly, vertical smearing is applied from the top of the image to the bottom and then the same procedure is repeated from bottom to top. This operation enables to fill the spaces for inner parts of each character. For vertical smearing, threshold value is chosen as 70. And this value is also determined with the help of approximate limit values of plate characters and test results. And, finally horizontal smearing is applied with a threshold value of 6. The program segment for these smearing algorithms is here as;

```
%Vertical Smearing from top to bottom
t=0;
c=0;
for y=1:size(plate,2)-1
    for x=1:size(plate,1)-1
```

```
            if plate(x,y)==1
            t=t+1;
            else
             t=0;
            c=0;
            end
                   if t<70
                   plate(x,y)=0;
                           else if t>=70 && c==0
                                  for k=t-69:t-1
                                  plate(k,y)=1;
                                  c=1;
                                  end
                            t=0;
                            end
                      end
            end
     end


%Vertical Smearing from bottom to top
t=0;
c=0;
for y=size(plate,2)-1:-1:1
      for x=size(plate,1)-1:-1:1
            if plate(x,y)==1
            t=t+1;
            else
            t=0;
            c=0;
            end
                  if t<70
                 plate(x,y)=0;
                          else if t>=70 && c==0
```

```
                                for k=t-69:t-1
                                plate(k,y)=1;
                                c=1;
                                end
                        t=0;
                        end
                end
        end
 end

%horizontal smearing for # of pixels smaller than 6
t=0;
c=0;
for x=1:size(plate,1)-1
t=0;
c=0;
        for y=1:size(plate,2)-1
              if plate(x,y)==1
              t=t+1;
              else
              t=0;
              c=0;
              end
                    if t<6
                    plate(x,y)=0;
                            else if t>=6 && c==0
                                    for k=y-5:y-1
                                    plate(x,k)=1;
                                    c=1;
                                    end
                              end
                        end
              end
    end
```

The resulting plate after smearing algorithms is shown in Figure 4.9. To see the overall performance of segmentation, the input for segmentation part, the image after filtering and dilation operations and output image after segmentation are given, respectively.

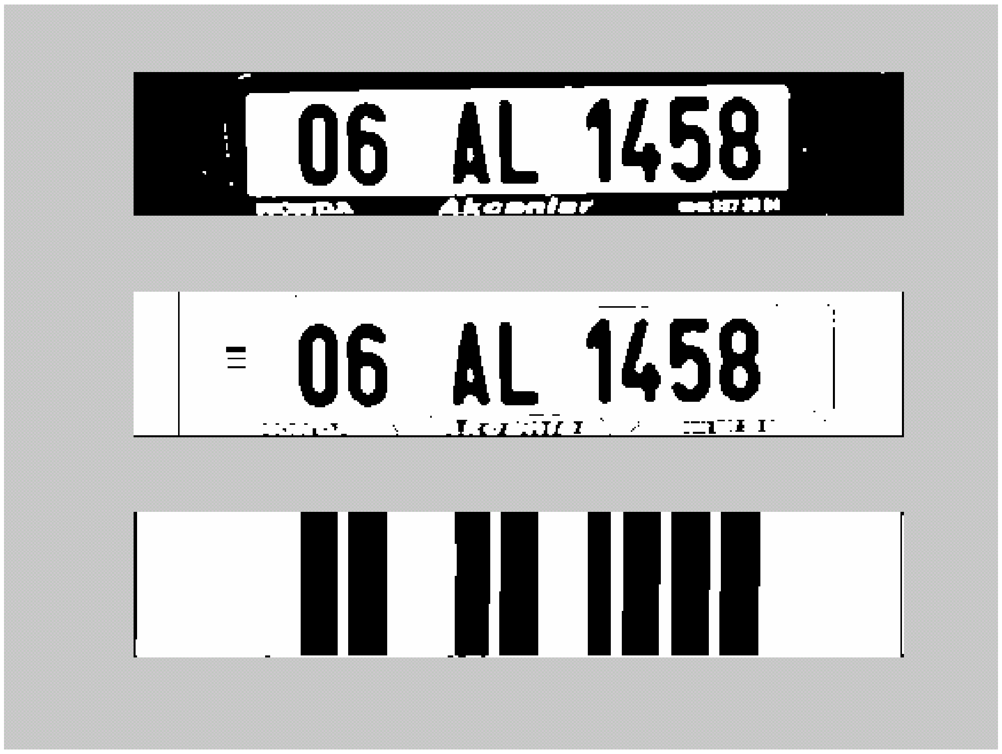


Figure 4.9: Plate images showing segmentation

Now, the locations of plate characters are known. These are labeled as black pixels and separated from each other. Next step is to cut the plate characters. It is done by finding starting and end points of characters in horizontal direction. Again the transitions of pixels from white to black or black to white are useful information to know that characters start and stop in these boundary points. The necessary program segment for this purpose is as follows;

```
%Finding starting and end points of characters
c=1;
r=1;
for y=50:size(character,2)-50
```

```
        if r~=character(1,y)
        m(c)=y;
        c=c+1;
        r=character(1,y);
        end
end

s=1;
for x=1:length(m)/2
      if m(s+1)-m(s)>=11
      m(s)=m(s);
      m(s+1)=m(s+1);
      s=s+2;
      else
      m(s)=2;
      m(s+1)=2;
      s=s+2;
      end
end
```

Although there are only plate characters in plate image, some points or regions may still remain. To eliminate this, some tests are applied to the found points that they are related with characters or not. And finally, these plate characters are cut using Matlab “imcrop” function taking the found starting and end points as an arguments.

The characters that are cut can be shown in Figure 4.10.

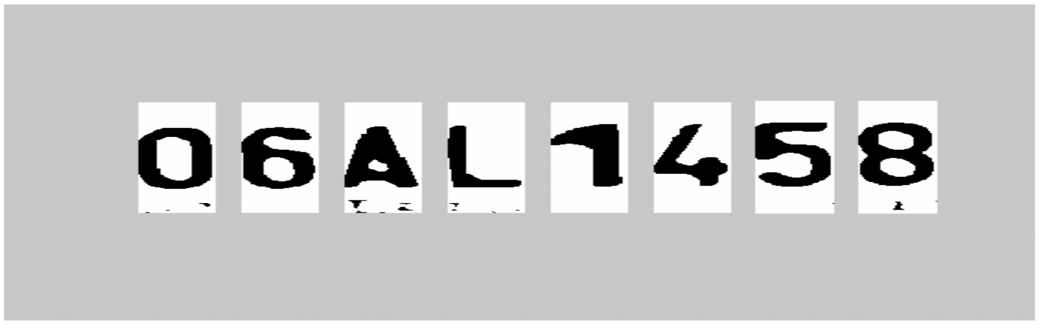


Figure 4.10: Individual characters

4.4 Recognition of Plate Characters

As mentioned in chapter 4, there are two main character recognition methods: template-based recognition and feature-based recognition. In this work, template-based recognition algorithm is used.

In template-based algorithm, the character image is compared with the ones in the database known as template and the best similarity is measured. This algorithm is named as template-matching. As mentioned in chapter 4, to measure the similarity and find the best match, a statistical method correlation is used. This method measures the correlation coefficient between a number of known images with the same size unknown images or parts of an image with the highest correlation coefficient between the images producing the best match. Since there are two signals as unknown input image and known database image in this system, cross-correlation is used.

Before recognition algorithm with template-matching, the characters must be normalized. Normalization is to refine the characters into a block that contains no extra white spaces (pixels) in all four sides of the characters. Then, each character is refined as shown in Figure 4.11.

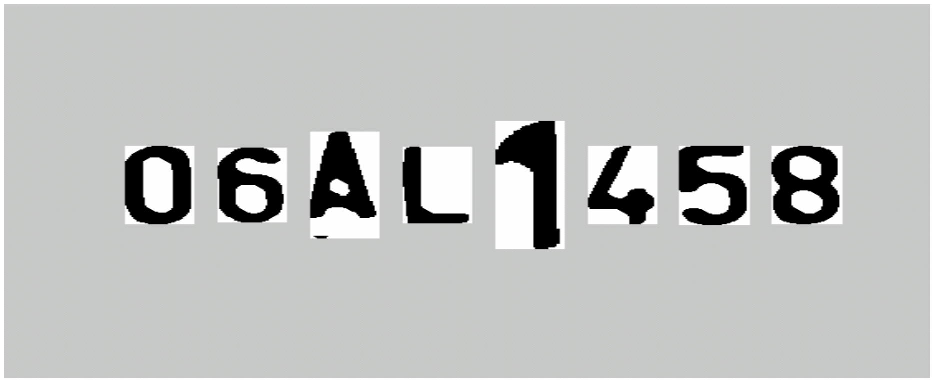


Figure 4.11: Refined characters

In Figure 4.11, it is obviously seen that characters sizes are different form each other. Therefore, character sizes must be fit to equal size. In other words, all character sizes must be equal-sized. This operation is made using Matlab “imresize” function. This function resizes an image of any type using the specified interpolation method. In this work, default interpolation, nearest neighbor interpolation, is used and the desired size values are given as an argument. Here the characters are fit to equal-sized as 36×18. Now, the characters become as in Figure 4.12.

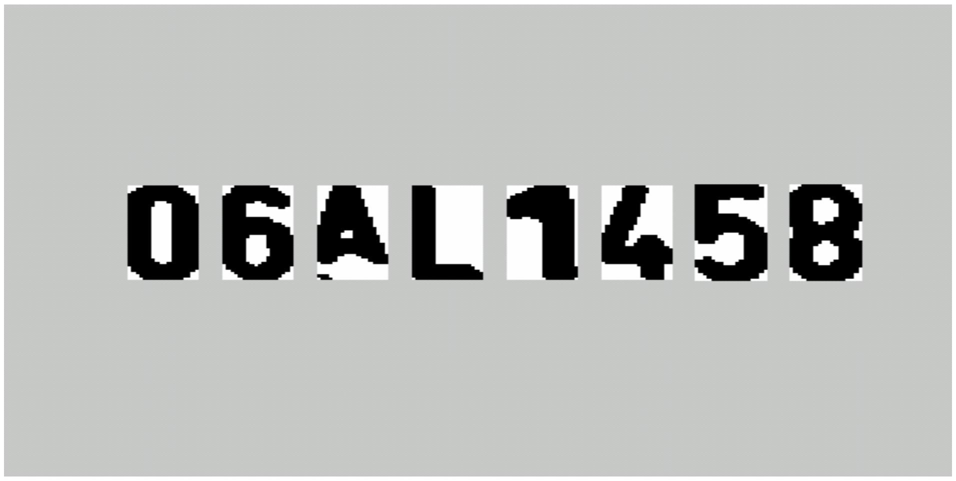


Figure 4.12: Equal-sized characters

Fitting approach is necessary for template matching. For matching the characters with the database, input images must be equal-sized with the database characters. Here the characters are fit to 36×18. The extracted characters cut from plate and the characters on database are now equal-sized. The next step is to form templates for matching algorithm.

The templates are necessary for matching. These templates are made as the matrices whose dimensions are the same as the plate characters that have been normalized and equal-sized. A sample for the letter B is shown in Figure 4.13.

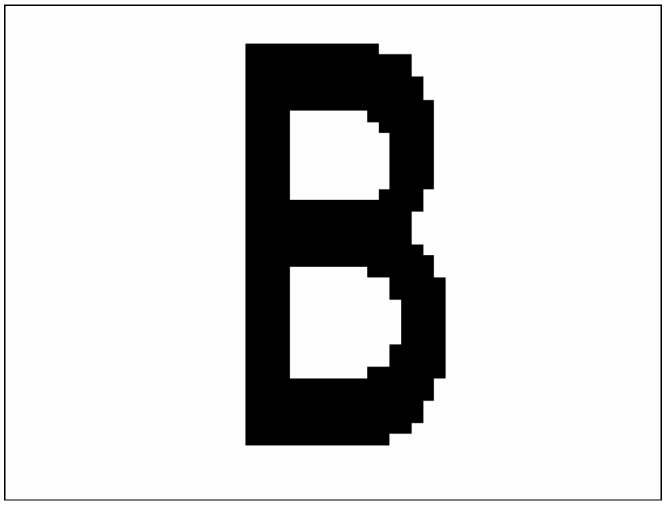


Figure 4.13: A sample template, letter B

And the collection of these templates for all characters is named as the database for this system. This system used the database as the Turkish license plates characters all 33 alphanumeric characters (23 alphabets and 10 numerals) with the size of 36×18. The database formed is shown in Figure 4.14.

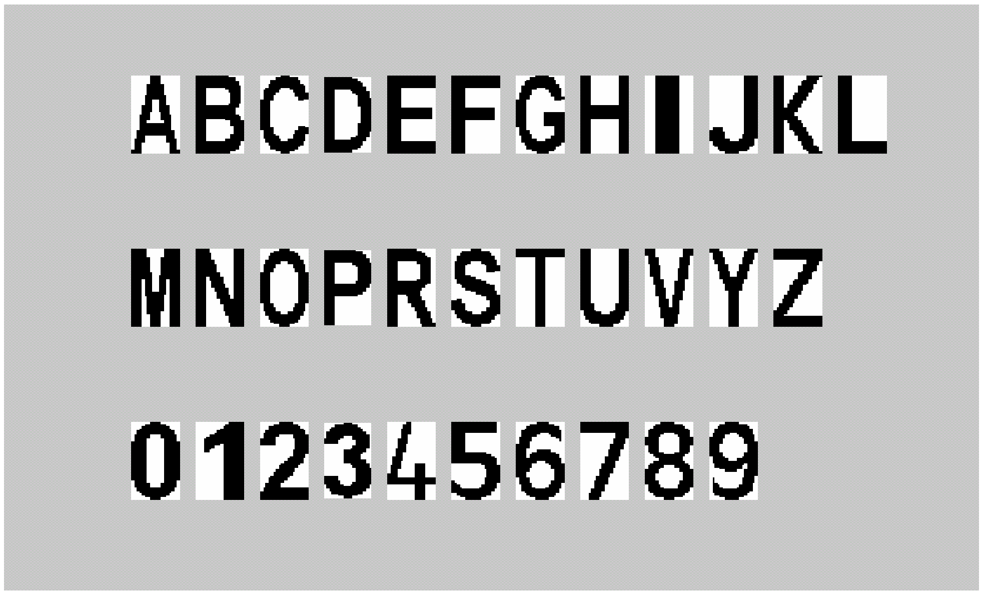


Figure 4.14: The database characters

Now, we have an input, plate characters and the templates known as database. The last step is to match the input with the templates and to find the best match. The cross-correlation function is used for this matching. This is done using Matlab “corr2” function. This function computes 2-D correlation coefficient between an input image and a template image, where both input and template images are matrices or vectors of the same size.

For an input, the necessary algorithm for calculating the correlation coefficients and finding the best similarity (the biggest correlation coefficient) is as follows;

```
% calculation the correlation coefficient for character1
t=1;
for d=1:18:594
w=corr2(character1,database(1:36,d:d+17));
w2(t)=w
t=t+1;
end
```

```
[y1,i(1)]=max(w2);


% calculation the correlation coefficient for character2
t=1;
for d=1:18:594
w=corr2(character2,database(1:36,d:d+17));
w2(t)=w
t=t+1;
end

[y2,i(2)]=max(w2);
```

This algorithm is repeated for all characters that have been obtained from the license plate. And the biggest correlation coefficients for all characters individually are recorded and these values are compared with the database. Finally, these values show the best similarity between the input character and the template that are most correlated. So, the value that is showing the related template is read as a recognition output.

Because of the similarities of some characters, there may be some errors during recognition. In other words, the algorithm can recognize the wrong letter or number. The confused characters mainly are B and 8, E and F, D and O, S and 5, Z and 2. To eliminate such mistakes and to increase the recognition rate, some criteria tests are used in the system for the confused characters defining the special features of the characters. Therefore, if one of these characters is detected, another checking algorithm is used to identify the character. With these features of characters and applied checking tests during recognition algorithm, recognition rate is increased with the minimum error.

CHAPTER 5

EXPERIMENTAL WORKS

As mentioned before, this license plate recognition system is designed for the identification Turkish license plates. And the test vehicle images containing Turkish license plates are given as a tester and the performance of the system are obtained.

The input vehicle images are captured by a digital camera. Two different cameras are used for the input images. Both the cameras have the same resolution value which is 1200x1600. And, the captured images are colored images. The test images are taken under various illumination conditions. In other words, the images are taken on the different time periods of the day with different light conditions. Also, the captured images are taken from 4-5 meters away from the vehicles.

The system is tested with a Pentium III computer 733 MHz with 128 MB RAM. The operating system is Windows 98. Moreover, the performance of the system is tested with a Windows 2000 operating system and with 256 MB RAM as well. And the system is designed in Matlab 6.5. Matlab is the software produced by The MathWorks Company and it is used for performing mathematical calculations, analyzing and visualizing data, and writing new software programs.

5.1 Accuracy Calculation

To calculate the performance of any system is primary concern for understanding the structure and measuring the accuracy. Determining the accuracy of an Automatic Vehicle Identification (AVI) system is a bit complex and it depends on the application that is applied, operating conditions and assumptions that is made during testing process.

An AVI system performance is difficult to quantify. It is encouraging to expect a system or a machine to be perfect and also it is encouraging to assume 100 percent accuracy in identifying license plates. On the other hand, some plates couldn't be read at all by a eye, human being, or by a machine system owing to damage, dirtiness of plate or obscuration. Therefore, an automatic system for vehicle identification shouldn't be expected to achieve perfect performance, even under ideal conditions such as weather, illumination, etc.

One of the methods to measure the reliability and success of license plate recognition system is as the percentage of license plates correctly identified by the system that may then be verified by a person observing the raw video signal on a monitor. If a person estimates the vehicle with its license plate from a poor video image, it is probable that the system can also produce a lower-confidence result. However, by looking at clear video image, the person is less likely to make mistake. And, an automated machine or system will also return to a higher degree of accuracy in a similar way.

An automated system can only identify or recognize the alphanumeric content of license plate after the system is well-arranged, meaning that the plate of the vehicle is in the field of view.

It is important to note that the overall accuracy of an automated system cannot be estimated directly from its individual characters accuracies. For instance, if we have a system that recognizes and identifies 10000 license plates with seven characters on each plate, 70000 characters totally. And, assuming that the system reads the six characters correctly on each plate, on the other hand, the system makes a mistake that one character is not recognized correctly on each plate. In other words, the system misses a character. From this point of view, one might be inclined to state the overall recognition as, 6 characters on each plate times totally 10000 license plates giving the 60000 plate characters to be recognized correctly. It can be formulized as;

6 characters x 10000 license plates = 60000 (recognized-characters)

And the overall accuracy of the system is expressed the ratio of recognized-characters to total number license plates. As a percentage, this can be formulized as;

Accuracy = (total number of recognized-characters / total number of plates) x 100%

For this example, the overall accuracy is;

Overall accuracy = (60000 / 70000) x 100 % = 85.71 %

However, true accuracy of the system in the example above is not 85.71 percent. Because, there isn't any correct plate recognition. All plate recognitions fail due to missing one character on each plate. Therefore, the true accuracy of the system in the example is zero.

Because of this, the overall system performance or accuracy must be corrected. So, the true overall system accuracy can be expressed as a formula given below;

$$\text{Accuracy}(A) = \left(\frac{\sum_{i} A_i}{T}\right) \times 100\% \tag{5.1}$$

where A_i is the recognition rate for each plate

T is the total number of license plates

and

$$A_i = (C_1 \times C_2 \times C_3 \times \ldots \times C_n) \tag{5.2}$$

where C_n is the rate of successful interpretation of nth character and C_n takes only one or zero values:

$C_n = 1 \Rightarrow$ for true recognition

$C_n = 0 \Rightarrow$ for false recognition.

This approach is the best recognition rate. On the other hand, this formulation or calculation shows only the performance of recognition part of a license plate recognition system. But, the system consists of three main parts: Extraction of plate region, segmentation of plate characters and recognition of characters. Therefore, the accuracies of all these parts of the system must be considered and all accuracies must be taken into consideration.

To find the overall performance of the system, it is important to determine the all accuracy rates for all parts of the system. So, the overall system performance can be defined as the product of all parts accuracy rates: Accuracy rate for extraction of plate region part, accuracy rate for segmentation of plate characters part and accuracy rate for recognition of characters part [46].

This definition or calculations can be formulized as given below;

$$\begin{aligned} \text{Recognition Rate of LPR System} &= \prod (\text{A1 x A2 x A3}) \\ &= \prod(\text{Percentages of Accuracies}) \end{aligned} \tag{5.3}$$

where,

A1 = recognition rate for extraction part

A2 = recognition rate for segmentation part

A3 = recognition rate for recognition part

For performance calculation, it is important to note that the system should be defined clearly and precisely and also the conditions must be explained clearly under which the system achieved the stated accuracy rate.

In this study, the system is tested over a large number of vehicle images and the overall performance of the system is calculated considering all parts of the system.

5.2 Test Results

As stated before, the captured images are taken from 4-5 meters away from the vehicles and the images are taken on the different time periods of the day with different light conditions. And fixed images-motionless video images- are used. As a summary, experiments have been performed to test the proposed system and to measure the accuracy of the system.

For the test, 340 different images for 340 different vehicles are used. For all input images, the system algorithms are applied and all the outputs of parts of the system are observed.

The results of the tests can be seen in Table 5.1.

Table 5.1: Results of the tests

UNITS OF LPR SYSTEM	NUMBER OF ACCURACY	PERCENTAGE OF ACCURACY
Extraction of Plate Region	332 / 340	97.65 %
Segmentation of Plate Characters	327 / 340	96.18 %
Recognition of Plate Characters	336 / 340	98.82 %

To observe the performance of the extraction of plate region part, the input images are processed with the algorithm and it was found that the exact location for vehicle plates are obtained successfully in 332 vehicle images. And in 8 vehicle images, the exact locations are not uniquely found. One failure for a test image can be shown in Figures 5.1, 5.2 and 5.3.

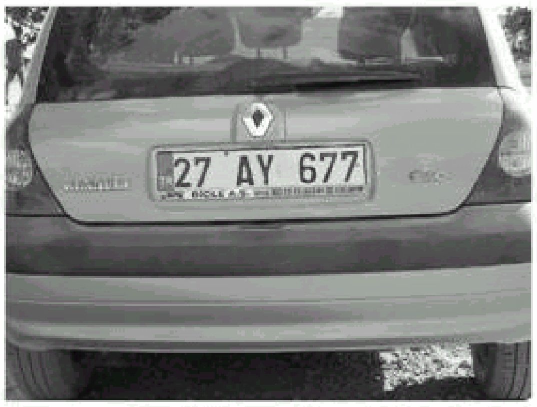


Figure 5.1: Input image for the test

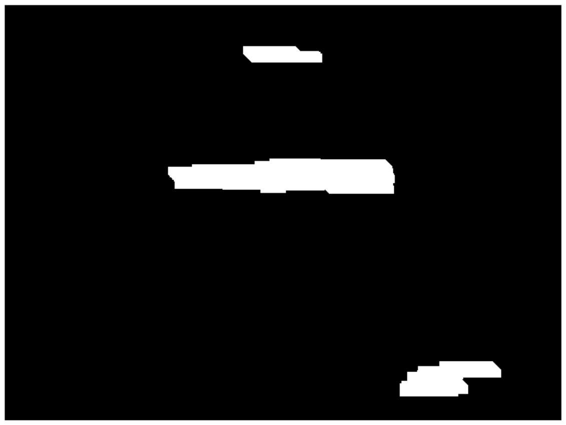


Figure 5.2: The output of extraction part
where more than one candidate region exist

Figure 5.3: The images showing the wrong region labeled after extraction

If these images are examined, it is seen that there are three candidate regions for plate location, not one. As mentioned in chapter 4, if there exists such a situation, some criteria tests are applied to the image. These tests are applied to the image. But, unfortunately the exact place of license plate couldn't be found.

After observing the extraction part, next step is to find the correctness of segmentation part of the system. It is clearly seen that the least accuracy rate or success is this part in this system. 327 of 340 input images are correctly segmented, and 13 images are not segmented correctly. The main failure occurs in the point that Turkish license plates have the international TR label on left side of the plate. In some plates this label is very close to the first character in the plate, so the segmentation cannot be fully achieved. The first character and the TR label are sensed as one character on a plate. An example of this situation can be observed in Figure 5.4, Figure 5.5 and Figure 5.6.

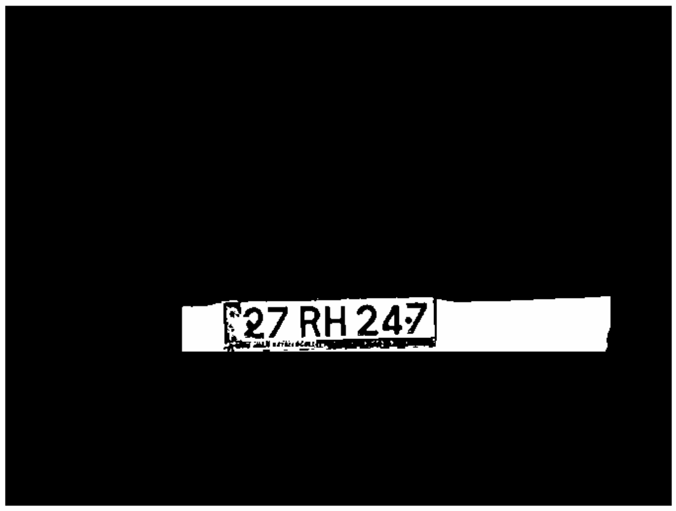


Figure 5.4 : A sample binarized image

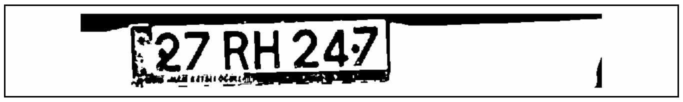


Figure 5.5 : Output image after extraction

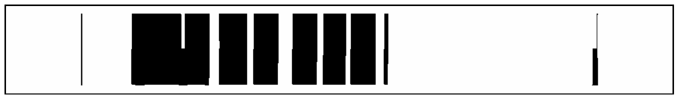


Figure 5.6: The plate image after segmentation process

It is seen from Figure 5.6, the first character 2 and the label TR is inseparable from each other. So, there is a segmentation failure here.

Similar failure may occur if the plate characters are written very close to each other. Also, the same failure may arise when the characters are written in bold-style. These situations may result such an effect.

And if the plates contain some dirt or spots on it, these undesired effects may cause some difficulties for separating the characters correctly. Therefore, these undesired noises must be filtered using effective filters.

Finally, the performance of the character recognition is to be searched. According to the values given in Table 5.1, the biggest accuracy rate is for recognition part, approximately 98.8 %. The character recognition algorithm depends on the correlations between the plate characters and the template characters. The best similarity between the input image and templates can be interpreted as true recognition.

This approach is very useful in many conditions. However, there are some failures or wrong recognition during this template matching algorithm. Because of the similarities for some characters, the recognition may fail. This undesired situation appears when the proposed recognition algorithm is tested. In other words, some characters may be confused during recognition part because of the similarities of some characters. The confused characters mainly are B and 8, Z and 7, D and 0, S and 5, Z and 2. To increase the recognition rate, some criteria tests are used in the system for the confused characters defining the special features of the characters. With the help of these unique properties of characters and applied tests during recognition algorithm, recognition rate is increased with the minimum error.

An example of such a failure can be shown for the letter B and the numeral 8. Let's assume that one of the characters in a license plate is B. And, also assume that, the system recognizes this character as a numeric character 8 instead of letter B (recognition failure). If these characters are examined in details, the unique properties can be revealed. To see the differences between the two characters, the template characters for the letter B the numeral 8 are shown in Figure 5.7.

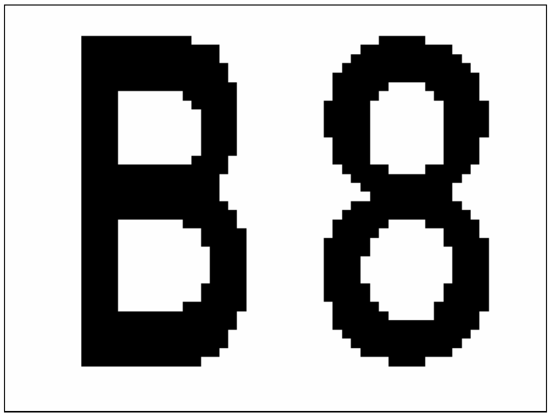


Figure 5.7: Sample templates for confusion of characters

For example, the letter B has only black pixels in first column of its matrix representation. However, the numeric character 8 hasn't. This is the difference on letter B with respect to numeral 8. This unique property can be used to detect the letter B from the numeral 8. If the system recognizes one of two characters as a true recognition, a checking algorithm is applied to the input character whether this is true or not. And, this property is used as a checking criterion.

During the tests of the system, "B", "A", "D", "S", and "Z" are confused with "8", "4", "0", "5" and "7" respectively.

Also, these confusions are more likely to be solved because one of the characters is a letter while the other one is a numeral. In considering the Turkish license plates, the numerals can be used at the beginning of the license plates or at the end of the license plates. Therefore, by checking the place of the characters, these confusions can be solved. For example, first two characters and last two characters must be a numeral or third character must be a letter. If these are thought as checking criteria, some confusion may be eliminated.

In this study, the plates are assumed parallel to the ground. This is because the camera that captures vehicle images is fixed at a point. So, there is no such an image that is not parallel to the ground. But, if an image exists that the captured image is not parallel to the ground, some additional image processing algorithms can be applied to the image to make the image parallel to the ground. A solution for this situation can be an algorithm that uses the Hough transform. An approach for this situation is explained in Appendix B.

As a summary, from the tests results seen in Table 5.1, the overall performance of the automatic vehicle identification system can be stated as the product of all recognition rates in each step, extraction of plate region, segmentation of plate characters and recognition of plate characters.

Recognition Rate of LPR System = ∏(Percentages of Accuracies)

=[(recognition rate for extraction part) x (recognition rate for segmentation part) x

(recognition rate for recognition part)]

Recognition Rate = 97.65 % x 96.18 % x 98.82 %

Recognition Rate = 92.80 %

If the recognition rate is thought as a ratio of the value at which the true recognition occurs to the total number of test images, the overall system performance becomes as;

Recognition Rate

= (total number of true recognized-images / total number of input images) x 100 %

So, the overall system performance is;

Recognition Rate = (321 / 340) x 100 % = 94.41 %

This value can be seen that it is bigger rate for the value presented before. However, as stated earlier, true performance of overall system depends on the all system units. Therefore, they must be appreciated independently. And overall system performance can be found as the product of all parts accuracies.

The failure of the system can be also defined as the error for each part of the system. And the error is defined as the ratio of the total number of faulty (wrong) output images to the total number of input images after each step. Therefore,

Error for extraction part = 8 / 340 x 100% = 2.35 %
Error for segmentation part = 13 / 340 x 100% = 3.82 %
Error for recognition part = 4 / 340 x 100% = 1.18 %

CONCLUSIONS

In this thesis, we presented a new algorithm for Automatic Vehicle Identification. Identification is made using the vehicle’s license plates. A system identifies the vehicle by recognizing the license plate automatically. And the system is designed for Turkish license plates.

Input of the system is the image of a vehicle captured by a camera. The captured image taken from 4-5 meters away is processed through the license plate extractor with giving its output to segmentation part. Segmentation part separates the characters individually. And finally recognition part recognizes the characters giving the result as the plate number.

For observing the performance of the system, 340 vehicle images are used which are taken with a digital camera at the car parks of Gaziantep University. The images are taken on different time periods of the day and also these test images were taken under various illumination conditions. The system is tested over a large number of images. Finally it is proved to be %97.65 for the extraction of plate region, %96.18 for the segmentation of the characters and %98.82 for the recognition unit accurate, giving the overall system performance %92.80 recognition rates. If the recognition rate is thought as a ratio of the value at which the true recognition occurs to the total number of test images, the overall system performance becomes as 94.41 percent. On the other hand, as mentioned earlier, true performance of overall system depends on the all system units. Therefore, they must be appreciated independently. And overall system performance can be found as the product of all parts accuracies.

Considering the extraction of plate region, the failure appears when there is more than one candidate region for plate location. This can be eliminated using some checking algorithms that are sensitive to the special properties of plate location. This operation makes the failure to the minimum.

During the segmentation part of the algorithm, the main failure can occur to segment or separate the first character in a plate. Since Turkish license plates have the international TR label on left side of the plate and this label may be very close to the first character in some plates, so the segmentation cannot be fully achieved. The first character and the TR label are sensed as one character on a plate. To solve such a failure, the plate image is dilated using morphological dilation operation. This is effective to isolate the characters from each other. And similar failure may be encountered when the characters are close to each other or characters are written in bold style. In segmentation, there may be some difficulties for separating the characters if the plates contain some dirt or spots on it, these undesired effects can be eliminated with filtering the plate image. Averaging filtering or low-pass filtering can be used for this noise suppression conditions.

The confusions between some characters are the most encountered problems of the character recognition algorithm. This is because of the similarities between some characters. The most recognition failures are between the “B” and “8”, “A” and “4”, “D” and “0”, “S” and “5”, and “Z” and “7”.

In order to moderate the errors or confusions, some criteria tests are used in the system for the confused characters defining the special features of the characters. With the help of these unique properties of characters and the checking algorithms giving the answer of true recognition, recognition rate is increased with the minimum error.

The shadow on the plate is another factor that reduces the recognition rate. If the plate is placed in a deeper way to the car, the sunlight may cause the shadow to appear on the plate. To use a camera that is sensitive to the illumination conditions and adaptive for these changes or to use a light source with the camera can minimize the effect of shade.

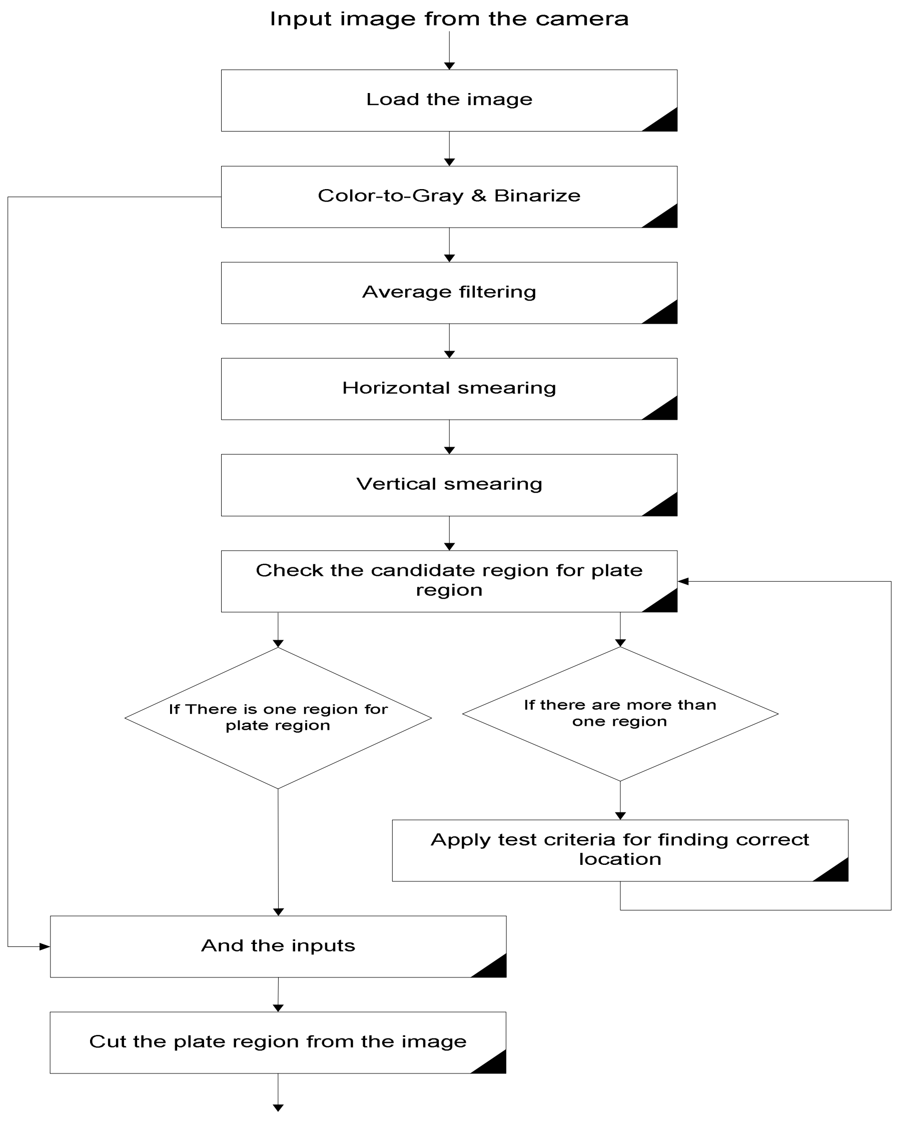
The proposed algorithm in this thesis works well on the vehicle images whose plates are clear enough to extract the characters and identify the vehicle automatically. And the recognition rate is good for the purpose.

During the efforts in studying this system, we have seen that the processing for identification takes a bit long time. The identification time can be reduced in future works. The other future work is to improve the recognition rate. Some different algorithms can be tried and the overall performance can be increased.

When dealing with recognition of Turkish license plates, it is thought that this system can be applied for other international plates. The first thing is to try to redesign the system for identification of multinational car license plates in future studies.

APPENDIX A

FLOW DIAGRAM OF THE ALGORITHM


Input image from the camera
Load the image
Color-to-Gray & Binarize
Average filtering
Horizontal smearing
Vertical smearing
Check the candidate region for plate region
If There is one region for plate region
If there are more than one region
Apply test criteria for finding correct location
And the inputs
Cut the plate region from the image

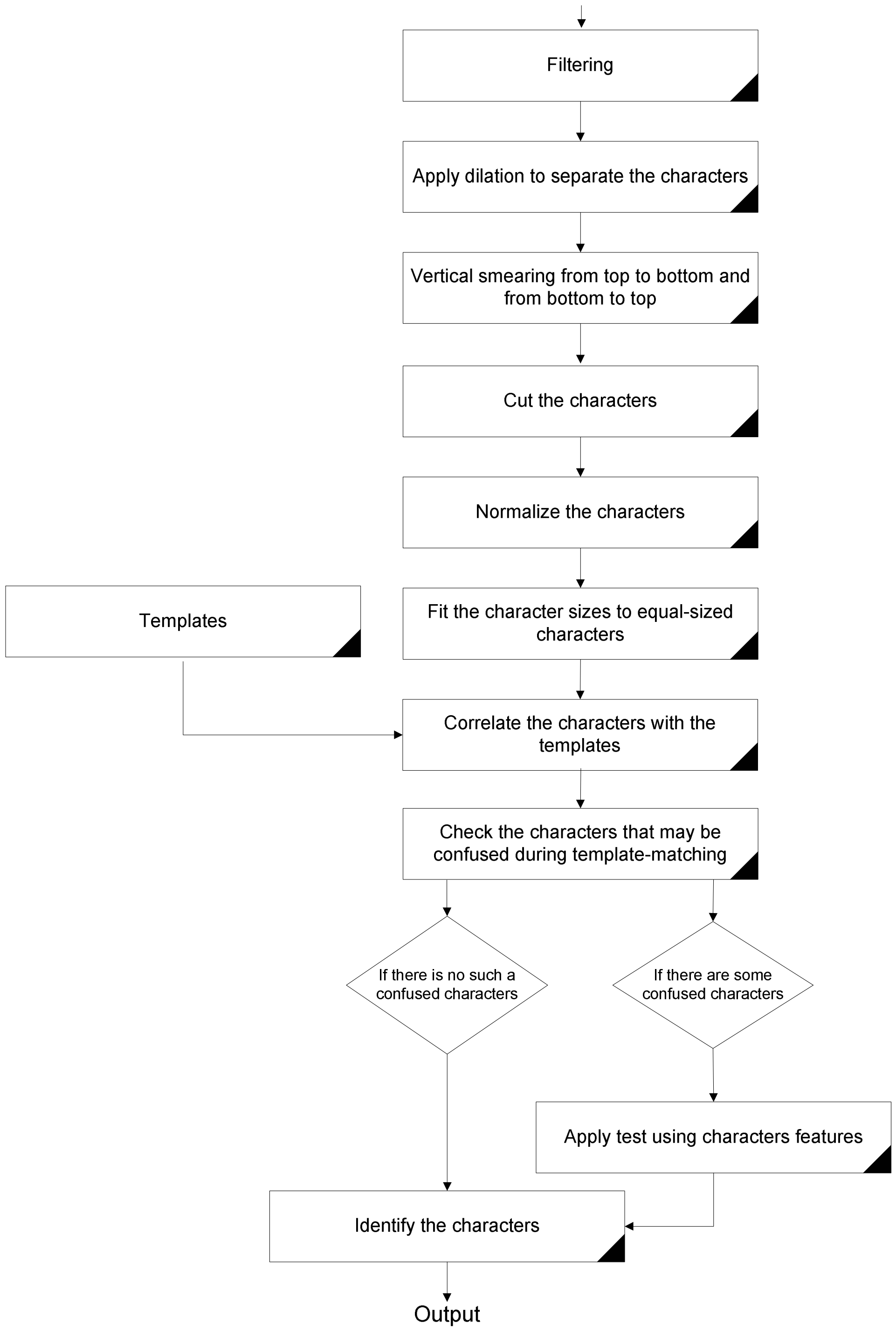

Filtering
Apply dilation to separate the characters
Vertical smearing from top to bottom and from bottom to top
Cut the characters
Normalize the characters
Templates
Fit the character sizes to equal-sized characters
Correlate the characters with the templates
Check the characters that may be confused during template-matching
If there is no such a confused characters
If there are some confused characters
Apply test using characters features
Identify the characters
Output

Figure A.1: Flow diagram of the algorithm

APPENDIX B

SKEW CORRECTION USING HOUGH TRANSFORM

In some situations, the vehicle images cannot be parallel to the ground. In other words, the license plate that is desired to recognize can be inclined. This situation is not appropriate for identification. Because, this can produce a failure during the recognition process of plate characters with template matching algorithm. Template matching algorithm uses the templates for comparing with the input and the templates are formed as parallel to the ground. If the characters are skew, the recognition process can give a wrong result.

To overcome this situation, Hough transform can be used. As mentioned earlier in chapter 3, Hough transform can be used to detect the lines in an image. And, if the lines are not parallel to the ground, this can be understood using this algorithm. An example of such a vehicle image is shown in Figure B.1. Also, the image after binarization process can be seen in Figure B.2 to see the inclination of the plate.

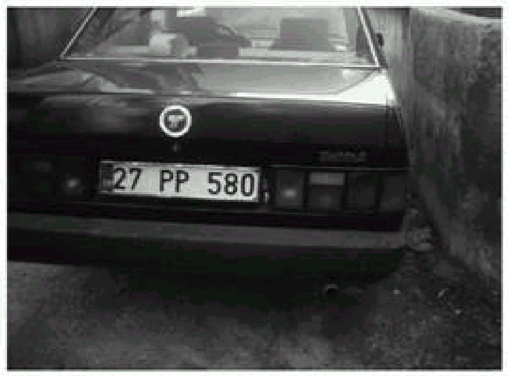


Figure B.1: A sample image for skew correction

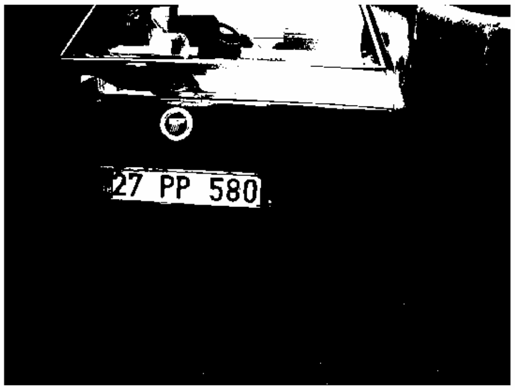


Figure B.2: The binarized image

If the lines are found firstly, and then the angle of the lines with respect to the ground is estimated, the image can be rotated in counter direction with this angle value. Namely, this operation normalizes the vehicle image with making it parallel to the ground.

To do this, an edge detection algorithm can be applied to the image to determine the lines. For doing it, Matlab “edge” function can be used. Edge function finds edges in intensity image. Edge function takes a binary image as its input, and returns a binary image of the same size as the input image, with 1's where the function finds edges in I and 0's elsewhere. The result of this application for the sample image given in Figure B.1 can be shown in Figure B.3.

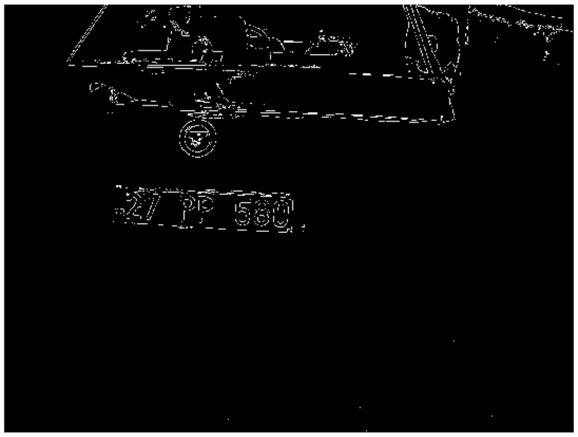


Figure B.3: Processed image after edge function

After that, dilation operation is done to make the lines or edges more visible. The result of it is in Figure B.4 below.

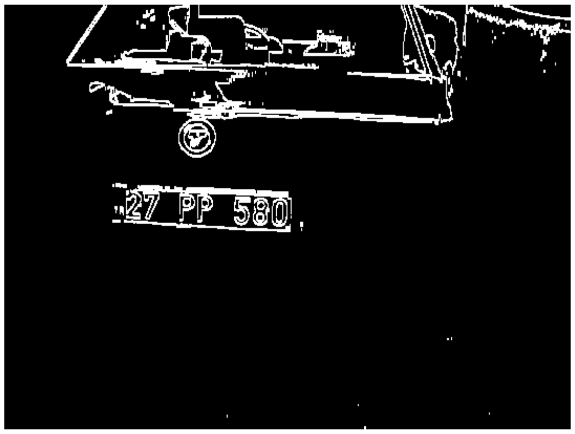


Figure B.4: The image after dilation operation

Now, an algorithm for detecting the lines and finding the angles of lines is used. For the purpose, an algorithm based on Hough transform is as follows;

```
% This function detects lines in a binary image using common computer vision
% operation known as the Hough Transform.
% Function uses Standard Hough Transform to detect Lines in a binary image.
% Arguments:
% Imbinary - a binary image. image pixels that have value equal to 1 are
% interested pixels for HOUGHLINE function.
% pstep  - interval for radius of lines in polar coordinates.
% tetastep - interval for angle of lines in polar coordinates.
% thresh   - a threshold value that determines the minimum number of pixels that
% belong to a line in image space.
% Returns:
% pdetect  - a vactor that contains radius of detected lines in polar coordinates
% system.
% tetadetect  - a vector that contains angle of detected lines in polar coordinates
% system.

function[pdetect,tetadetect,] = houghline(Imbinary,pstep,tetastep,thresh)

p = 1:pstep:sqrt((size(Imbinary,1))^2+(size(Imbinary,2))^2);
teta = 0:tetastep:180-tetastep;

Accumulator = zeros(length(p),length(teta));
[yIndex xIndex] = find(Imbinary);
for cnt = 1:size(xIndex)
   Indteta = 0;
   for tetai = teta*pi/180
      Indteta = Indteta+1;
      roi = xIndex(cnt)*cos(tetai)+yIndex(cnt)*sin(tetai);
      if roi >= 1 & roi <= p(end)
         temp = abs(roi-p);
         mintemp = min(temp);
         Indp = find(temp == mintemp);
         Indp = Indp(1);
         Accumulator(Indp,Indteta) = Accumulator(Indp,Indteta)+1;
      end
```

```
    end
end

AccumulatorbinaryMax = imregionalmax(Accumulator);
[Potential_p Potential_teta] = find(AccumulatorbinaryMax == 1);
Accumulatortemp = Accumulator - thresh;
pdetect = [];tetadetect = [];
for cnt = 1:length(Potential_p)
    if Accumulatortemp(Potential_p(cnt),Potential_teta(cnt)) >= 0
        pdetect = [pdetect;Potential_p(cnt)];
        tetadetect = [tetadetect;Potential_teta(cnt)];
    end
end

% Calculation of detected lines parameters(Radius & Angle).
pdetect = pdetect * pstep;
tetadetect = tetadetect *tetastep - tetastep;
```

This function is written in Matlab. And it is useful for skew correction. If the angle value of the lines can be found, then the image can be rotated to normalize the image in opposite direction. This rotation operation can be made using Matlab “imrotate” function.

APPENDIX C

FUNCTION DESCRIPTIONS

This part provides detailed descriptions of the functions that are used in this thesis. These are the Matlab functions in the Image Processing Toolbox. It begins with a list of functions grouped by subject area and continues with the detailed explanations of the functions in alphabetical order.

Image File I/O

- imread : Read image file

- imwrite : Write image file

Geometric Operations

- imcrop : Crop image

- imresize : Resize image

- imrotate : Rotate image

Pixel Values and Statistics

- corr2 : Compute 2-D correlation coefficient

- imhist : Display histogram of image data

Image Analysis

- edge : Find edges in intensity image

Image Enhancement

- histeq : Enhance contrast using histogram equalization

- imadjust : Adjust image intensity values

Linear Filtering

- filter2 : Perform 2-D filtering

- fspecial : Create predefined filters

Binary Image Operations

- dilate : Perform dilation on binary image

- erode : Perform erosion on binary image

Image Types and Type Conversions

- im2bw : Convert image to binary image by thresholding

- rgb2gray : Convert RGB image or colormap to grayscale

CORR2

Purpose:

Compute the two-dimensional correlation coefficient between two matrices

Syntax:

r = corr2(A,B)

Description:

r = corr2(A,B) computes the correlation coefficient between A and B, where A and B are matrices or vectors of the same size.

Class Support:

A and B can be of class double or of any integer class. r is a scalar of class double.

Algorithm:

corr2 computes the correlation coefficient using

$$r=\frac{\sum_m\sum_n(A_{mn}-\bar{A})(B_{mn}-\bar{B})}{\sqrt{\left[\sum_m\sum_n(A_{mn}-\bar{A})^2\right]\left[\sum_m\sum_n(B_{mn}-\bar{B})^2\right]}} \tag{C.1}$$

where $\bar{A}=\text{mean2}(A)$, and $\bar{B}=\text{mean2}(B)$

DILATE

Purpose:

Perform dilation on a binary image

Syntax:

BW2 = dilate(BW1,SE)

BW2 = dilate(BW1,SE,alg)

BW2 = dilate(BW1,SE,...,n)

Description:

BW2 = dilate(BW1,SE) performs dilation on the binary image
BW1, using the binary structuring element SE. SE is a matrix containing only 1's and 0's.

BW2 = dilate(BW1,SE,alg) performs dilation using the specified algorithm. alg is a string that can have one of these values:

•'spatial' (default) – processes the image in the spatial domain.

•'frequency' – processes the image in the frequency domain.

Both algorithms produce the same result, but they make different tradeoffs between speed and memory use. The frequency algorithm is faster for large images and structuring elements than the spatial algorithm, but uses much more memory.

BW2 = dilate(BW1,SE,...,n) performs the dilation operation n times.

Class Support:

The input image BW1 can be of class double or uint8. The output image BW2 is of class uint8.

Remarks:

You should use the frequency algorithm only if you have a large amount of memory on your system. If you use this algorithm with insufficient memory, it may actually be slower than the spatial algorithm, due to virtual memory paging. If the frequency algorithm slows down your system excessively, or if you receive "out of memory" messages, use the spatial algorithm instead.

<u>EDGE</u>

Purpose:

Find edges in an intensity image

Syntax:

```
BW = edge(I,'sobel')
BW = edge(I,'sobel',thresh)
BW = edge(I,'sobel',thresh,direction)
[BW,thresh] = edge(I,'sobel',...)

BW = edge(I,'prewitt')
BW = edge(I,'prewitt',thresh)
BW = edge(I,'prewitt',thresh,direction)
[BW,thresh] = edge(I,'prewitt',...)

BW = edge(I,'roberts')
BW = edge(I,'roberts',thresh)
[BW,thresh] = edge(I,'roberts',...)

BW = edge(I,'log')
BW = edge(I,'log',thresh)
BW = edge(I,'log',thresh,sigma)
[BW,threshold] = edge(I,'log',...)

BW = edge(I,'zerocross',thresh,h)
[BW,thresh] = edge(I,'zerocross',...)

BW = edge(I,'canny')
BW = edge(I,'canny',thresh)
BW = edge(I,'canny',thresh,sigma)
[BW,threshold] = edge(I,'canny',...)
```

Description:

edge takes an intensity image I as its input, and returns a binary image BW of the same size as I, with 1's where the function finds edges in I and 0's elsewhere.

edge supports six different edge-finding methods:

•The Sobel method finds edges using the Sobel approximation to the derivative. It returns edges at those points where the gradient of I is maximum.

•The Prewitt method finds edges using the Prewitt approximation to the derivative. It returns edges at those points where the gradient of I is maximum.

•The Roberts method finds edges using the Roberts approximation to the derivative. It returns edges at those points where the gradient of I is maximum.

•The Laplacian of Gaussian method finds edges by looking for zero crossings after filtering I with a Laplacian of Gaussian filter.

•The zero-cross method finds edges by looking for zero crossings after filtering I with a filter you specify.

•The Canny method finds edges by looking for local maxima of the gradient of I. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be “fooled” by noise, and more likely to detect true weak edges.

The parameters you can supply differ depending on the method you specify. If you do not specify a method, edge uses the Sobel method.

Class Support:

I can be of class uint8, uint16, or double. BW is of class uint8.

Remarks:

For the 'log' and ‘zerocross' methods, if you specify a threshold of 0, the output image has closed contours, because it includes all of the zero crossings in the input image.

<u>ERODE</u>

Purpose:

Perform erosion on a binary image

Syntax:

BW2 = erode(BW1,SE)

BW2 = erode(BW1,SE,alg)

BW2 = erode(BW1,SE,...,n)

Description:

BW2 = erode(BW1,SE) performs erosion on the binary image
BW1, using the binary structuring element SE. SE is a matrix containing only 1's and 0's.

BW2 = erode(BW1,SE,alg) performs erosion using the specified algorithm. alg is a string that can have one of these values:

•'spatial' (default) – processes the image in the spatial domain.

•'frequency' – processes the image in the frequency domain.

Both algorithms produce the same result, but they make different tradeoffs between speed and memory use. The frequency algorithm is faster for large images and structuring elements than the spatial algorithm, but uses much more memory.

BW2 = erode(BW1,SE,...,n) performs the erosion operation n times.

Class Support:

The input image BW1 can be of class double or uint8. The output image BW2 is of class uint8.

Remarks:

You should use the frequency algorithm only if you have a large amount of memory on your system. If you use this algorithm with insufficient memory, it may actually be slower than the spatial algorithm, due to virtual memory paging. If the frequency algorithm slows down your system excessively, or if you receive "out of memory" messages, use the spatial algorithm instead.

<u>FILTER2</u>

Purpose:

Perform two-dimensional linear filtering

Syntax:

B = filter2(h,A)

B = filter2(h,A,shape)

Description:

B = filter2(h,A) filters the data in A with the two-dimensional FIR filter in the matrix h. It computes the result, B, using two dimensional correlation, and returns the central part of the correlation that is the same size as A.

B = filter2(h,A,shape) returns the part of B specified by the shape parameter. shape is a string with one of these values:

•'full' returns the full two-dimensional correlation. In this case, B is larger than A.

•'same' (the default) returns the central part of the correlation. In this case, B is the same size as A.

•'valid' returns only those parts of the correlation that are computed without zero-padded edges. In this case, B is smaller than A.

Class Support:

The matrix inputs to filter2 can be of class double or of any integer class. The output matrix B is of class double.

Remarks:

Two-dimensional correlation is equivalent to two-dimensional convolution with the filter matrix rotated 180 degrees.

<u>FSPECIAL</u>

Purpose:

Create predefined filters

Syntax:

h = fspecial(type)

h = fspecial(type,parameters)

Description:

h = fspecial(type) creates a two-dimensional filter h of the specified type. (fspecial returns h as a computational molecule, which is the appropriate form to use with filter2.) type is a string having one of these values:

•'gaussian' for a Gaussian lowpass filter
•'sobel' for a Sobel horizontal edge-emphasizing filter
•'prewitt' for a Prewitt horizontal edge-emphasizing filter
•'laplacian' for a filter approximating the two-dimensional Laplacian operator
•'log' for a Laplacian of Gaussian filter
•'average' for an averaging filter
•'unsharp' for an unsharp contrast enhancement filter

Depending on type, fspecial may take additional parameters which you can supply. These parameters all have default values.

h = fspecial('gaussian',n,sigma) returns a rotationally symmetric Gaussian lowpass filter with standard deviation sigma (in pixels). n is a 1-by-2 vector specifying the number of rows and columns in h. (n can also be a scalar, in which case h is n-by-n.) If you do not specify the parameters, fspecial uses the default values of [3 3] for n and 0.5 for sigma.

h = fspecial('sobel') returns this 3-by-3 horizontal edge-finding and y-derivative approximation filter:

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

To find vertical edges, or for x-derivatives, use –h'.

h = fspecial('prewitt') returns this 3-by-3 horizontal edge-finding and y-derivative approximation filter:

```
[ 1    1    1
  0    0    0
 –1   –1   –1 ]
```

To find vertical edges, or for x-derivatives, use –h'.

h = fspecial('laplacian',alpha) returns a 3-by-3 filter approximating the two-dimensional Laplacian operator. The parameter alpha controls the shape of the Laplacian and must be in the range 0 to 1.0. fspecial uses the default value of 0.2 if you do not specify alpha.

h = fspecial('log',n,sigma) returns a rotationally symmetric Laplacian of Gaussian filter with standard deviation sigma (in pixels). n is a 1-by-2 vector specifying the number of rows and columns in h. (n can also be a scalar, in which case h is n-by-n.) If you do not specify the parameters, fspecial uses the default values of [5 5] for n and 0.5 for sigma.

h = fspecial('average',n) returns an averaging filter. n is a 1-by-2 vector specifying the number of rows and columns in h. (n can also be a scalar, in which case h is n-by-n.) If you do not specify n, fspecial uses the default value of [3 3].

h = fspecial('unsharp',alpha) returns a 3-by-3 unsharp contrast enhancement filter. fspecial creates the unsharp filter from the negative of the Laplacian filter with parameter alpha. alpha controls the shape of the Laplacian and must be in the range 0 to 1.0 fspecial uses the default value of 0.2 if you do not specify alpha.

<u>HISTEQ</u>

Purpose:

Enhance contrast using histogram equalization

Syntax:

```
J = histeq(I,hgram)
J = histeq(I,n)
[J,T] = histeq(I,...)

newmap = histeq(X,map,hgram)
newmap = histeq(X,map)
[newmap,T] = histeq(X,...)
```

Description:

histeq enhances the contrast of images by transforming the values in an intensity image, or the values in the colormap of an indexed image, so that the histogram of the output image approximately matches a specified histogram.

J = histeq(I,hgram) transforms the intensity image I so that the histogram of the output intensity image J with length(hgram) bins approximately matches hgram. The vector hgram should contain integer counts for equally spaced bins with intensity values from 0 to 1.0. histeq automatically scales hgram so that sum(hgram) = prod(size(I)). The histogram of J will better match hgram when length(hgram) is much smaller than the number of discrete levels in I.

J = histeq(I,n) transforms the intensity image I, returning in J an intensity image with n discrete gray levels. A roughly equal number of pixels is mapped to each of the n levels in J, so that the histogram of J is approximately flat. (The histogram of J is flatter when n is much smaller than the number of discrete levels in I.) The default value for n is 64.

[J,T] = histeq(I,...) returns the gray scale transformation that maps gray levels in the intensity image I to gray levels in J.

newmap = histeq(X,map,hgram) transforms the colormap associated with the indexed image X so that the histogram of the gray component of the indexed image (X,newmap) approximately matches hgram. histeq returns the transformed colormap in newmap. length(hgram) must be the same as size(map,1).

newmap = histeq(X,map) transforms the values in the colormap so that the histogram of the gray component of the indexed image X is approximately flat. It returns the transformed colormap in newmap.

[newmap,T] = histeq(X,...) returns the grayscale transformation T that maps the gray component of map to the gray component of newmap.

Class Support:
For syntaxes that include an intensity image I as input, I can be of class uint8, uint16, or double, and the output image J has the same class as I. For syntaxes that include an indexed image X as input, X can be of class uint8 or double; the output colormap is always of class double. Also, the optional output T (the gray level transform) is always of class double.

<u>**IM2BW**</u>

Purpose:
Convert an image to a binary image, based on threshold

Syntax:
BW = im2bw(I,level)
BW = im2bw(X,map,level)
BW = im2bw(RGB,level)

Description:
im2bw produces binary images from indexed, intensity, or RGB images. To do this, it converts the input image to grayscale format (if it is not already an intensity image), and then converts this grayscale image to binary by thresholding. The output binary image BW has values of 0 (black) for all pixels in the input image with luminance less than level and 1 (white) for all other pixels. (Note that you specify level in the range [0,1], regardless of the class of the input image.)

BW = im2bw(I,level) converts the intensity image I to black and white.

BW = im2bw(X,map,level) converts the indexed image X with colormap map to black and white.

BW = im2bw(RGB,level) converts the RGB image RGB to black and white.

Class Support:
The input image can be of class uint8, uint16, or double. The output image BW is of class uint8.

<u>IMADJUST</u>

Purpose:
Adjust image intensity values or colormap

Syntax:
J = imadjust(I,[low high],[bottom top],gamma)
newmap = imadjust(map,[low high],[bottom top],gamma)
RGB2 = imadjust(RGB1,...)

Description:
J = imadjust(I,[low high],[bottom top],gamma) transforms the values in the intensity image I to values in J by mapping values between low and high to values between bottom and top. Values below low and above high are clipped; that is, values below low map to bottom, and those above high map to top. You can use an empty matrix ([]) for [low high] or for [bottom top] to specify the default of [0 1]. gamma specifies the shape of the curve describing the relationship between the values in I and J. If gamma is less than 1, the mapping is weighted toward higher (brighter) output values. If gamma is greater than 1, the mapping is weighted toward lower (darker) output values. If you omit the rgument, gamma defaults to 1 (linear mapping).

newmap = imadjust(map,[low high],[bottom top],gamma) ransforms the colormap associated with an indexed image. If [low high] and [bottom top] are both 2-by-3, and gamma is a 1- by-3 vector, imadjust rescales the red, green, and blue components separately. The rescaled colormap, newmap, is the same size as map.

RGB2 = imadjust(RGB1,...) performs the adjustment on each image plane (red, green, and blue) of the RGB image RGB1. As with the colormap adjustment, you can use different parameter values for each plane by specifying [low high] and [bottom top] as 2-by-3 matrices, and gamma as a 1-by-3 vector.

Class Support:

For syntax that include an input image (rather than a colormap), the image can be of class uint8, uint16, or double. The output image is of the same class as the input image. For syntax that include a colormap, the input and output colormaps are of class double.

Remarks:

If top < bottom, the output image is reversed (i.e., as in a negative).

<u>IMCROP</u>

Purpose:

Crop an image

Syntax:

I2 = imcrop(I)

X2 = imcrop(X,map)

RGB2 = imcrop(RGB)

I2 = imcrop(I,rect)

X2 = imcrop(X,map,rect)

RGB2 = imcrop(RGB,rect)

[...] = imcrop(x,y,...)

[A,rect] = imcrop(...)

[x,y,A,rect] = imcrop(...)

Description:

imcrop crops an image to a specified rectangle. In the syntaxes below, imcrop displays the input image and waits for you to specify the crop rectangle with the mouse.

```
I2 = imcrop(I)
X2 = imcrop(X,map)
RGB2 = imcrop(RGB)
```

If you omit the input arguments, imcrop operates on the image in the current axes.
To specify the rectangle:
•For a single-button mouse, press the mouse button and drag to define the crop rectangle. Finish by releasing the mouse button.
•For a 2- or 3-button mouse, press the left mouse button and drag to define the crop rectangle. Finish by releasing the mouse button.
If you hold down the Shift key while dragging, or if you press the right mouse button on a 2- or 3-button mouse, imcrop constrains the bounding rectangle to be a square. When you release the mouse button, imcrop returns the cropped image in the supplied output argument. If you do not supply an output argument, imcrop displays the output image in a new figure.

Class Support:

The input image A can be of class uint8, uint16, or double. The output image B is of the same class as A. rect is always of class double.

Remarks:

Because rect is specified in terms of spatial coordinates, the width and height elements of rect do not always correspond exactly with the size of the output image. For example, suppose rect is [20 20 40 30], using the default spatial coordinate system. The upper-left corner of the specified rectangle is the center of the pixel (20,20) and the lower-right corner is the center of the pixel (50,60). The resulting output image is 31-by-41, not 30-by-40, because the output image includes all pixels in the input image that are completely or partially enclosed by the rectangle.

IMHIST

Purpose:

Display a histogram of image data

Syntax:

imhist(I,n)

imhist(X,map)

[counts,x] = imhist(...)

Description:

imhist(I,n) displays a histogram with n bins for the intensity image I above a grayscale colorbar of length n. If you omit the argument, imhist uses a default value of n = 256 if I is a grayscale image, or n = 2 if I is a binary image.

imhist(X,map) displays a histogram for the indexed image X. This histogram shows the distribution of pixel values above a colorbar of the colormap map. The colormap must be at least as long as the largest index in X. The histogram has one bin for each entry in the colormap.

[counts,x] = imhist(...) returns the histogram counts in counts and the bin locations in x so that stem(x,counts) shows the histogram. For indexed images, it returns the histogram counts for each colormap entry; the length of counts is the same as the length of the colormap.

Class Support:

The input image can be of class uint8, uint16, or double.

IMREAD

Purpose:

Read images from graphics files

Syntax:

A = imread(filename,fmt)

[X,map] = imread(filename,fmt)

[...] = imread(filename)

[...] = imread(...,idx) (TIFF only)

[...] = imread(...,ref) (HDF only)

[...] = imread(...,'BackgroundColor',BG) (PNG only)

[A,map,alpha] = imread(...) (PNG only)

Description:

A = imread(filename,fmt) reads a grayscale or truecolor image named filename into A. If the file contains a grayscale intensity image, A is a two-dimensional array. If the file contains a truecolor (RGB) image, A is a three-dimensional (m-by-n-by-3) array.

[X,map] = imread(filename,fmt) reads the indexed image in filename into X and its associated colormap into map. The colormap values are rescaled to the range [0,1]. A and map are two-dimensional arrays.

[...] = imread(filename) attempts to infer the format of the file from its content. filename is a string that specifies the name of the graphics file, and fmt is a string that specifies the format of the file. If the file is not in the current directory or in a directory in the MATLAB path, specify the full pathname for a location on your system. If imread cannot find a file named filename, it looks for a file named filename.fmt. If you do not specify a string for fmt, the toolbox will try to discern the format of the file by checking the file header.

Imread can read BMP, HDF, JPEG, PCX, PNG, TIFF and XVD.

Class Support:

In most of the image file formats supported by imread, pixels are stored using eight or fewer bits per color plane. When reading such a file, the class of the output (A or X) is uint8. imread also supports reading 16-bit-per-pixel data from TIFF and PNG files; for such image files, the class of the output (A or X) is uint16. Note that for indexed images, imread always reads the colormap Into an array of class double, even though the image array itself may be of class uint8 or uint16.

IMRESIZE

Purpose:
Resize an image

Syntax:
B = imresize(A,m,method)
B = imresize(A,[mrows ncols],method)
B = imresize(...,method,n)
B = imresize(...,method,h)

Description:
imresize resizes an image of any type using the specified Interpolation method. method is a string that can have one of these values:
•'nearest' (default) uses nearest neighbor interpolation.
•'bilinear' uses bilinear interpolation.
•'bicubic' uses bicubic interpolation.

If you omit the method argument, imresize uses the default method of 'nearest'.

Class Support:
The input image can be of class uint8, uint16, or double. The output image is of the same class as the input image.

IMROTATE

Purpose:
Rotate an image

Syntax:

B = imrotate(A,angle,method)

B = imrotate(A,angle,method,'crop')

Description:

B = imrotate(A,angle,method) rotates the image A by angle degrees in a counter-clockwise direction, using the specified interpolation method. Method is a string that can have one of these values:

•'nearest' (default) uses nearest neighbor interpolation.

•'bilinear' uses bilinear interpolation.

•'bicubic' uses bicubic interpolation.

If you omit the method argument, imrotate uses the default method of 'nearest'.
The returned image matrix B is, in general, larger than A to include the whole rotated image. imrotate sets invalid values on the periphery of B to 0.

B = imrotate(A,angle,method,'crop') rotates the image A through angle degrees and returns the central portion which is the same size as A.

Class Support:

The input image can be of class uint8, uint16, or double. The output image is of the same class as the input image.

Remarks:

To rotate the image clockwise, specify a negative angle.

<u>**IMWRITE**</u>

Purpose:

Write an image to a graphics file

Syntax:

imwrite(A,filename,fmt)

imwrite(X,map,filename,fmt)

imwrite(...,filename)

imwrite(...,Param1,Val1,Param2,Val2...)

Description:

imwrite(A,filename,fmt) writes the image in A to filename. A can be either a grayscale image (M-by-N) or a truecolor image (M-by-N-by-3). If A is of class uint8 or uint16, imwrite writes the actual values in the array to the file. If A is of class double, imwrite rescales the values in the array before writing, using uint8(round(255*A)). This operation converts the floating-point numbers in the range [0,1] to 8-bit integers in the range [0,255].

imwrite(X,map,filename,fmt) writes the indexed image in X and its associated colormap map to filename. If X is of class uint8 or uint16, imwrite writes the actual values in the array to the file. If X is of class double, imwrite offsets the values in the array before writing using uint8(X–1). Map must be a valid MATLAB colormap of class double; imwrite rescales the values in map using uint8(round(255*map)). Note that most image file formats do not support colormaps with more than 256 entries.

Class Support:

Most of the supported image file formats store uint8 data. PNG and TIFF additionally support uint16 data. For grayscale and RGB images, if the data array is double, the assumed dynamic range is [0,1]. The data array is automatically scaled by 255 before being written out as uint8. If the data array is uint8 or uint16 (PNG and TIFF only), then it is written out without scaling as uint8 or uint16, respectively.

<u>**RGB2GRAY**</u>

Purpose:

Convert an RGB image or colormap to grayscale

Syntax:

I = rgb2gray(RGB)

newmap = rgb2gray(map)

Description:

rgb2gray converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

I = rgb2gray(RGB) converts the truecolor image RGB to the grayscale intensity image I.

newmap = rgb2gray(map) returns a grayscale colormap equivalent to map.

Class Support:

If the input is an RGB image, it can be of class uint8, uint16, or double; the output image I is of the same class as the input image. If the input is a colormap, the input and output colormaps are both of class double.

Algorithm:

rgb2gray converts the RGB values to NTSC coordinates, sets the hue and saturation components to zero, and then converts back to RGB color space.

REFERENCES

[1] Yuntao Cui and Qian Huang, (1997). Character Extraction of License Plates from Video, Conference on Computer Vision and Pattern Recognition (CVPR'97)

[2] K.K. Kim, K.I. Kim, J.B. Kim, H.J. Kim, (2000). Learning Based Approach For License Plate Recognition, Proceedings of the 2000 IEEE Signal Processing Society Workshop, Volume 2, pp. 614-623

[3] B. Martin, P. Scott, (1992). Automatic Vehicle Identification: A Test of Theories of Technology Published in Science, Technology & Human Values, Volume 17, No. 4, pp. 485-505,

[4] Morrison, A. Steven, (1986). A Survey of Road Pricing, Transportation Research A, Volume 20, No.2, pp. 87-97

[5] Dawson, J. A. L. (1983). Electronic road pricing in Hong Kong: the pilot stage, Traffic Engineering + Control, Volume 24, pp. 372-374.

[6] I. Catling, Bob McQueen, (1991). Road transport informatics in Europe major programs and demonstrations, IEEE Transactions on Vehicular Technology, Volume 40, pp. 132-140.

[7] W. J. Gillan, (1988). Prometheus-Reducing traffic congestion by advanced technology. In Roads and Traffic 2000, International Road and Traffic Conference, Volume 1, pp.111-115.

[8] Tsuzawa, Masami and H. Okamoto, (1998). Overview and perspective of Advanced Mobile Traffic Information & Communication System (AMTICS). In Roads and Traffic 2000, International Road and Traffic Conference, Volume 1, pp. 153-157.

[9] Kawashima, Hironao, (1991). Two major programs and demonstrations in Japan, IEEE Transactions on Vehicular Technology, Volume 40, pp. 141-146.

[10] H.J. Stoelhurst, and A. J. Zandbergen, (1990). The development of a road pricing system in the Netherlands, Traffic Engineering + Control, Volume 31, pp. 66-71.

[11] K. Waersted, and K. Bogen, (1989). No stop electronic toll payment systems, Second International Conference on Road Traffic Monitoring, pp. 128-132.

[12] P. Bohnke and E. Pfannerstill, (1986, January). A System for the Automatic Surveillance of Traffic Situations, Institute of Transportation Engineers Journal, pp. 41-45

[13] Rong-Shyang Ju, T. H. Maze, (1989, July/August). Freeway Surveillance and Control System Using Simulation Model, Journal of Transportation Engineering, Volume 115, No. 4, pp. 425-437

[14] Berka, Stanislaw and B. Kent Lall. (1998, Jan/Feb). New Perspectives for ATMS: Advanced Technologies in Traffic Detection, Journal of Transportation Engineering, pp. 9-15.

[15] A. T. Bergan, Loyd Henion, Milan Krukar and Brian Taylor, (1988). Electronic License Plate Technology: Automatic Vehicle Location and Identification', Canadian Journal of Civil Engineering, Vol. 15, No. 6, pp. 1035-1042

[16] Butterfield, Earl, and Mark Haselkorn and Kathy Alalusi. (1994, July). Potential of Automatic Vehicle Identification in the Puget Sound Area., Washington State Department of Transportation, Washington State Transportation Center, University of Washington

[17] Turner, Shawn M. (1996). Advanced Techniques for Travel Time Data Collection, Innovative Transportation Data Management, Survey Methods, and Geographic Information Systems/TRB, National Research Council Washington D.C.: National Academy Press, pp. 51-8.

[18] Burges CJC, Ben JI, Denker JS, Lecun Y, Nohl CR, (1993). Off line recognition of handwritten postal words using neural networks, International Journal of Pattern Recognition & Artificial Intelligence, Volume 7, No. 4, pp. 689-704.

[19] Camastra F, Vinciarelli A, (2001). Cursive character recognition by learning vector quantization, Pattern Recognition Letters, Volume 22, No. 6-7, pp.625-629

[20] Cortes C, Vapnik V, (1995). Support-vector Networks, Machine Learning, Volume 20, No.3, pp.273-297

[21] P. Hu, Y. Zhao, J. Wang, Z. Yang, (2002). Recognition of Gray Character using Gabor Filters, Proceedings of the Fifth International Conference on Information Fusion, Volume 1, pp. 419-424

[22] G. Chen, T. D. Bui, (1999). Invariant Fourier-wavelet descriptor for pattern recognition, Pattern Recognition, Volume 32, No.7

[23] W.L.Hwang, F. Chang, (1998). Character extraction from documents using wavelet maxima, Image and Vision Computing, Volume 16, No.5

[24] R. Porter, N.Canagarajah, (1997). Robust rotation-invariant texture classification:wavelet, Gabor fitler and GMRF based schemes, IEE Proceedings:Vision, Image & Signal Processing, Volume 144, No.3, pp. 180-188

[25] H. Chao, D. Jianyu, Y.F. Zheng, S.C. Ahalt, (2001). Object tracking using the Gabor wavelet transform and golden section algorithm, Proceedings 2001 ICRA, IEEE International Conference on Robotics and Robotics and Automation, Vol.2, pp. 1671-1676

[26] J.G. Daugman, (1980). Two-dimensional spectral analysis of cortical receptive field profiles, Vision Research, Vol.20, pp. 847-856

[27] V. Tavsanoglu, E. Saatci, (2000), Feature extraction for character recognition using Gabor-type filters implemented by cellular neural networks, Proceedings of the 2000 6th IEEE International Workshop on Cellular Neural Networks and their Applications, IEEE. 2000, pp. 63-8.

[28] H. Yoshimura, M. Etoh, K. Kondo, N.Yokoya N, (2000). Gray-scale character recognition by Gabor jets projection, Proceedings 15th International Conference on Pattern Recognition, ICPR-2000. IEEE Comput. Soc. Part Volume 2, pp.335-8

[29] R.A.Lotufo, A.D.Morgan, and A.S.Johnson, (1990). Automatic Number-Plate Recognition, Proceedings of the IEE Colloquium on Image analysis for Transport Applications, Volume 35, pp.6/1-6/6

[30] A.S. Johnson, B.M. Bird, (1990, April). Number-plate Matching for Automatic Vehicle Identification, IEE Colloquium on Electronic Image and Image Processing in Security and Forensic.

[31] M.M.M. Fahmy, (1994). Automatic Number-plate Recognition: Neural Network Approach, Proceedings of VNIS'94 Vehicle Navigation and Information System Conference, 3 1 Aug-2 Sept

[32] J.A.G. Nijhuis, M.H. Ter Brugge, K.A.Helmholt , J.P.W.Pluim, L. Spaanenburg, R.S. Venema, M.A.Westenberg, (1995). Car License Plate Recognition with Neural Networks and Fuzzy Logic, IEEE International Conference on Neural Networks.

[33] H.J. Choi, (1987). A Study on the Extraction and Recognition of a Car Number Plate by Image Processing, Journal of the Korea Institute of Telematics and Electronics, Volume 24, pp. 309-3 15

[34] H.S. Kim, et al., (1991). Recognition of a Car Number Plate by a Neural Network, Proceedings of the Korea Information Science Society Fall Conference, Volume 18, pp. 259-262

[35] E.R. Lee, P.K. Kim, and H.J. Kim, (1994). Automatic Recognition of a Car License Plate Using Color Image Processing, Proceedings of the International Conference on Image Processing

[36] S.K. Kim, D.W. Kim, and H.J. Kim, (1996). A Recognition of Vehicle License Plate Using a Genetic Algorithm Based Segmentation, Proceedings of 3rd IEEE International Conference on Image Processing, Volume 2, pp. 661-664

[37] H. Hontani, and T. Koga, (2001). Character extraction method without prior knowledge on size and information, Proceedings of the IEEE International Vehicle Electronics Conference (IVEC'01), pp. 67-72.

[38] S.H. Park, K.I. Kim, K. Jung, and H.J. Kim, (1999). Locating car license plates using neural network, IEE Electronics Letters, Volume 35, No. 17, pp. 1475-1477.

[39] H.J.Kim, D.W.Kim, S.K. Kim, J.V. Lee, J.K. Lee, (1997). Automatic Recognition of Car Licence Plates Using Color Image Processing, Engineering Design & Automation, 3(2), pp. 215-225

[40] Image Processing Fundamentals [Ebook]
http://www.ph.tn.tudelft.nl/Courses/FIP/frames/fip.html

[41] Gonzalez and Woods, (2002). Digital Image Processing, Prentice Hall (Second Edition)

[42] William K. Pratt, (2001). Digital Image Processing, John Wiley & Sons, Inc. (Third Edition)

[43] Hough Transform, http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm

[44] Character Recognition, http://en.wikipedia.org/wiki/Optical_character_recognition

[45] M.Horowitz, (1957). Efficient use of a picture correlator, J. Opt. Soc. Am, Volume 47, pp.327

[46] S. Ozbay, and E. Ercelebi, (2005). Automatic Vehicle Identification by Plate Recognition, Transactions on Enformatika System Sciences and Engineering, Volume 9, pp. 222-225.