



Journal Homepage: -www.journalijar.com
**INTERNATIONAL JOURNAL OF
ADVANCED RESEARCH (IJAR)**

Article DOI:10.21474/IJAR01/ 9492
 DOI URL: <http://dx.doi.org/10.21474/IJAR01/9492>



RESEARCH ARTICLE

AUTOMATIC VEHICLE NUMBER PLATE RECOGNITION.

Mrs. Vijeeta Patil.

Manuscript Info

Manuscript History

Received: 05 June 2019
 Final Accepted: 07 July 2019
 Published: August 2019

Key words:-

Thresholding ,edge detection,template matching,optical character recognition.

Abstract

Automatic number plate recognition is one of the techniques that can be used for the identification of vehicles number plate. The purpose of this project is to investigate a suitable way to recognize the registration plate from an image of vehicle.

Copy Right, IJAR, 2019,. All rights reserved.

Introduction:-

Automatic vehicle number plate recognition systems as a practical application of artificial Intelligence. Massive integration of information technologies into all aspects of modern life caused demand for processing vehicles as conceptual resources in information systems. Because a standalone information system without any data has no sense, there was also a need to transform information about vehicles between the reality and information systems. This can be achieved by a human agent, or by special intelligent equipment which is able to recognize vehicles by their number plates in a real environment and reflect it into conceptual resources. Because of this, various recognition techniques have been developed and number plate recognition systems are today used in various traffic and security applications, such as parking, access and border control, or tracking of stolen cars.

In parking, number plates are used to calculate duration of the parking. When a vehicle enters an input gate, number plate is automatically recognized and stored in database. When a vehicle later exits the parking area through an output gate, number plate is recognized again and paired with the first one stored in the database. The difference in time is used to calculate the parking fee. Automatic number plate recognition systems can be used in access control. For example, this technology is used in many companies to grant access only to vehicles of authorized personnel.

In some countries, ANPR systems installed on country borders automatically detect and monitor border crossings. Each vehicle can be registered in a central database and compared to a black list of stolen vehicles. In traffic control, vehicles can be directed to different lanes for a better congestion control in busy urban communications during the rush hours.

Chapter 1

Mathematical aspects of number plate recognition systems

In most cases, vehicles are identified by their number plates, which are easily readable for humans, but not for machines. For machine, a number plate is only a grey picture defined as a two-dimensional function $f(x,y)$, where x and y are spatial coordinates, and f is a light intensity at that point. Because of this, it is necessary to design robust mathematical machinery, which will be able to extract semantics from spatial domain of the captured image. These functions are implemented in so-called "ANPR systems", where the acronym "ANPR" stands for an "Automatic

Number Plate Recognition". ANPR system means transformation of data between the real environment and information systems.

The design of ANPR systems is a field of research in artificial intelligence, machine vision, pattern recognition and neural networks. Because of this, the main goal of this thesis is to study algorithmic and mathematical principles of automatic number plate recognition systems.

Physical aspects of number plate recognition systems

Automatic number plate recognition system is a special set of hardware and software components that precedes an input graphical signal like static pictures or video sequences, and recognizes license plate characters from it. A hardware part of the ANPR system typically consists of a camera, image processor, camera trigger, communication and storage unit. The hardware trigger physically controls a sensor directly installed in a lane. Whenever the sensor detects a vehicle in a proper distance of camera, it activates a recognition mechanism. Alternative to this solution is a software detection of an incoming vehicle, or continual processing of the sampled video signal. Software detection or continual video processing may consume more system resources, but it does not need additional hardware equipment, like the hardware trigger.

Image processor recognizes static snapshots captured by the camera, and returns a text representation of the detected license plate. ANPR units can have own dedicated image processors (all-in-one solution), or they can send captured data to a central processing unit for further processing (generic ANPR). The image processor is running on special recognition software, which is a key part of whole ANPR system. Because one of the fields of application is a usage on road lanes, it is necessary to use a special camera with the extremely short shutter. Otherwise, quality of captured snapshots will be degraded by an undesired motion blur effect caused by a movement of the vehicle. For example, usage of the standard camera with shutter of $1/100 \text{ sec}$ to capture a vehicle with speed of 80 km/h will cause a motion skew in amount of 0.22 m . This skew means the significant degradation of recognition abilities.

There is also a need to ensure system invariance towards the light conditions. Normal camera should not be used for capturing snapshots in darkness or night, because it operates in a visible light spectrum. Automatic number plate recognition systems are often based on cameras operating in an infrared band of the light spectrum. Usage of the infrared camera in combination with an infrared illumination is better to achieve this goal. Under the illumination, plates that are made from reflexive material are much more highlighted than rest of the image. This fact makes detection of license plates much easier.



Figure 1.1:-(a) Illumination makes detection of reflexive image plates easier. (b) Long Camera shutter and a movement of the vehicle can cause an undesired motion blur effect

Notations and mathematical symbols

Logic symbols

$p \square q$ Exclusive logical disjunction ($p \text{ xor } q$)

$p \sqcap q$ Logical conjunction ($p \text{ and } q$)

$p \sqcup q$ Logical disjunction ($p \text{ or } q$)

$\square p$ Exclusion (not p)

Mathematical definition of image

$f(x, y)$

x and y are spatial coordinates of an image, and f is an intensity of light at that point. This function is always discrete on digital computers.

Chapter 2**Literature survey**

The ANPR was invented in 1976 at the Police Scientific Development Branch in the UK. Prototype systems were working by 1979, and contracts were let to produce industrial systems, first at EMI Electronics.

Components

The software aspect of the system runs on standard PC hardware and can be linked to other applications or databases. It first uses a series of image manipulation techniques to detect, normalize and enhance the image of the number plate, and then optical character recognition (OCR) to extract the alpha numeric's of the license plate. ANPR systems are generally deployed in one of two basic approaches: one allows for the entire process to be performed at the lane location in real-time, and the other transmits all the images from many lanes to a remote computer location and performs the OCR process there at some later point in time. When done at the lane site, the information captured of the plate alphanumeric, date-time, lane identification, and any other information that is required is completed in somewhere around 250 milliseconds. This information, now small data packets, can easily be transmitted to some remote computer for further processing if necessary, or stored at the lane for later retrieval. In the other arrangement, there are typically large numbers of PCs used in a server farm to handle high workloads, such as those found in the London congestion charge project. Often in such systems, there is a requirement to forward images to the remote server, and this can require larger bandwidth transmission media.

Technology

ANPR uses optical character recognition (OCR) on images taken by cameras. When Dutch vehicle registration plates switched to a different style in 2002, one of the changes made was to the font, introducing small gaps in some letters (such as *P* and *R*) to make them more distinct and therefore more legible to such systems. Some license plate arrangements use variations in font sizes and positioning—ANPR systems must be able to cope with such differences in order to be truly effective. More complicated systems can cope with international variants, though many programs are individually tailored to each country.

The cameras used can include existing road-rule enforcement or closed-circuit television cameras, as well as mobile units, which are usually attached to vehicles. Some systems use infrared cameras to take a clearer image of the plate. Installing ANPR cameras on law enforcement vehicles requires careful consideration of the juxtaposition of the cameras to the license plates they are to read. Using the right number of cameras and positioning them accurately for optimal results can prove challenging, given the various missions and environments at hand. Highway patrol requires forward-looking cameras that span multiple lanes and are able to read license plates at very high speeds. City patrol needs shorter range, lower focal length cameras for capturing plates on parked cars. Parking lots with perpendicularly parked cars often require a specialized camera with a very short focal length. Most technically advanced systems are flexible and can be configured with a number of cameras ranging from one to four which can easily be repositioned as needed. States with rear-only license plates have an additional challenge since a forward-looking camera is ineffective with incoming traffic. In this case one camera may be turned backwards.

Algorithms

There are six primary algorithms that the software requires for identifying a license plate:

1. Plate localization – responsible for finding and isolating the plate on the picture.
2. Plate orientation and sizing – compensates for the skew of the plate and adjusts the dimensions to the required size.
3. Normalization – adjusts the brightness and contrast of the image.
4. Character segmentation – finds the individual characters on the plates.
5. Optical character recognition.

Syntactical/Geometrical analysis – check characters and positions against country-specific rules.

The complexity of each of these subsections of the program determines the accuracy of the system. During the third phase (normalization), some systems use edge detection techniques to increase the picture difference between the letters and the plate backing. A median filter may also be used to reduce the visual noise on the image.

Imaging Hardware

At the front end of any ANPR system is the imaging hardware which captures the image of the license plates. The initial image capture forms a critically important part of the ANPR system which, in accordance to the Garbage In, Garbage Out principle of computing, will often determine the overall performance.

License plate capture is typically performed by specialized cameras designed specifically for the task. Factors which pose difficulty for license plate imaging cameras include speed of the vehicles being recorded, varying ambient lighting conditions, headlight glare and harsh environmental conditions. Most dedicated license plate capture cameras will incorporate infrared illumination in order to solve the problems of lighting and plate reflectivity.

Many countries now use license plates that are retro reflective. This returns the light back to the source and thus improves the contrast of the image. In some countries, the characters on the plate are not reflective, giving a high level of contrast with the reflective background in any lighting conditions. A camera that makes use of active infrared imaging (with a normal color filter over the lens and an infrared illuminator next to it) benefits greatly from this as the infrared waves are reflected back from the plate. This is only possible on dedicated ANPR cameras, however, and so cameras used for other purposes must rely more heavily on the software capabilities. Further, when a full-color image is required as well as uses of the ANPR-retrieved details it is necessary to have one infrared-enabled camera and one normal (color) camera working together.

To maximize the chances of effective license plate capture, installers should carefully consider the positioning of the camera relative to the target capture area. Exceeding threshold angles of incidence between camera lens and license plate will greatly reduce the probability of obtaining usable images due to distortion. Manufacturers have developed tools to help eliminate errors from the physical installation of license plate capture cameras

Chapter 3**System requirements****System specification**

1. Software specification
2. Java 1.6
3. Java Net Beans
4. Windows 98

Hardware specification

1. Hard disk: 40 GB
2. RAM:128 mb
3. Processor: Pentium 4
4. Monitor: 15" color monitor
5. Floppy drive: 1.44 MB
6. Mouse :HCL
7. CD Drive: LG 52X
8. Printer: Laser

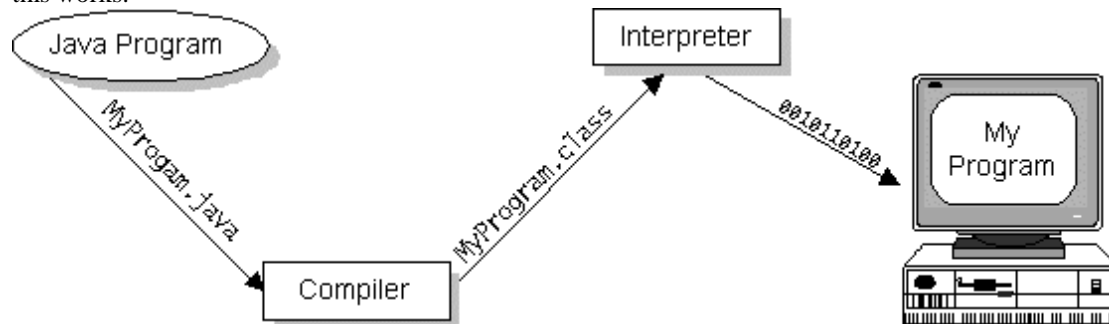
Software Description:**Java:**

Java was conceived by James Gosling, Patrick Naughton, Chris Wrath, Ed Frank, and Mike Sheridan at Sun Micro system. It is an platform independent programming language that extends its features wide over the network. Java2 version introduces an new component called "Swing" – is a set of classes that provides more power & flexible components than are possible with AWT.

1. It's a light weight package, as they are not implemented by platform-specific code.
2. Related classes are contained in javax.swing and its sub packages, such as javax.swing.tree.

3. Components explained in the Swing have more capabilities than those of AWT

Java is also unusual in that each Java program is both compiled and interpreted. With a compiler, you translate a Java program into an intermediate language called **Java byte codes**--the platform-independent codes interpreted by the Java interpreter. With an interpreter, each Java byte code instruction is parsed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. This figure illustrates how this works.



Execution process of .java file.

Java byte codes can be considered as the machine code instructions for the **Java Virtual Machine** (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make "write once, run anywhere" possible. The Java program can be compiled into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. For example, the same Java program can run on Windows NT, Solaris, and Macintosh.

The Java Platform

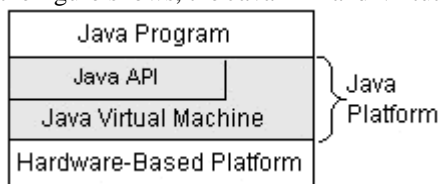
A platform is the hardware or software environment in which a program runs. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other, hardware-based platforms. Most other platforms are described as a combination of hardware and operating system.

The Java platform has two components:

The **Java Virtual Machine** (Java VM)

The **Java Application Programming Interface** (Java API)

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries (**packages**) of related components. The following figure depicts a Java program, such as an application or applet, that's running on the Java platform. As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.



As a platform-independent environment, Java can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring Java's performance close to that of native code without threatening portability.

What Can Java Do?

Probably the most well-known Java programs are **Java applets**. An applet is a Java program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, Java is not just for writing cute, entertaining applets for the World Wide Web ("Web"). Java is a general-purpose, high-level programming language and a powerful software platform. Using the generous Java API, we can write many types of programs.

The most common types of programs are probably applets and applications, where a Java application is a standalone program that runs directly on the Java platform.

How does the Java API support all of these kinds of programs?

With packages of software components that provide a wide range of functionality. The **core API** is the API included in every full implementation of the Java platform. The core API gives you the following features:

The Essentials: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.

Applets: The set of conventions used by Java applets.

Networking: URLs, TCP and UDP sockets, and IP addresses.

Internationalization: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.

Security: Both low-level and high-level, including electronic signatures, public/private key management, access control, and certificates.

Software components: Known as JavaBeans, can plug into existing component architectures such as Microsoft's OLE/COM/Active-X architecture, OpenDoc, and Netscape's Live Connect.

Object serialization: Allows lightweight persistence and communication via Remote Method Invocation (RMI).

Java Database Connectivity (JDBC): Provides uniform access to a wide range of relational databases.

Java not only has a core API, but also standard extensions. The standard extensions define APIs for 3D, servers, collaboration, telephony, speech, animation, and more.

We explore the `java.net` package, which provides support for networking. Its creators have called Java "programming for the Internet." These networking classes encapsulate the "socket" paradigm pioneered in the Berkeley Software Distribution (BSD) from the University of California at Berkeley.

Networking Basics

Ken Thompson and Dennis Ritchie developed UNIX in concert with the C language at Bell Telephone Laboratories, Murray Hill, New Jersey, in 1969. In 1978, Bill Joy was leading a project at Cal Berkeley to add many new features to UNIX, such as virtual memory and full-screen display capabilities. By early 1984, just as Bill was leaving to found Sun Microsystems, he shipped 4.2BSD, commonly known as Berkeley UNIX. 4.2BSD came with a fast file system, reliable signals, interprocess communication, and, most important, networking. The networking support first found in 4.2 eventually became the de facto standard for the Internet. Berkeley's implementation of TCP/IP remains the primary standard for communications with the Internet. The socket paradigm for inter process and network communication has also been widely adopted outside of Berkeley.

Socket Overview

A network socket is a lot like an electrical socket. Various plugs around the network have a standard way of delivering their payload. Anything that understands the standard protocol can "plug in" to the socket and communicate.

Internet protocol (IP) is a low-level routing protocol that breaks data into small packets and sends them to an address across a network, which does not guarantee to deliver said packets to the destination.

Transmission Control Protocol (TCP) is a higher-level protocol that manages to reliably transmit data. A third protocol, User Datagram Protocol (UDP), sits next to TCP and can be used directly to support fast, connectionless, unreliable transport of packets.

Client/Server

A server is anything that has some resource that can be shared. There are compute servers, which provide computing power; print servers, which manage a collection of printers; disk servers, which provide networked disk space; and web servers, which store web pages. A client is simply any other entity that wants to gain access to a particular server.

In Berkeley sockets, the notion of a socket allows a single computer to serve many different clients at once, as well as serving many different types of information. This feat is managed by the introduction of a port, which is a numbered socket on a particular machine. A server process is said to “listen” to a port until a client connects to it. A server is allowed to accept multiple clients connected to the same port number, although each session is unique. To manage multiple client connections, a server process must be multithreaded or have some other means of multiplexing the simultaneous I/O.

Reserved Sockets

Once connected, a higher-level protocol ensues, which is dependent on which port you are using. TCP/IP reserves the lower, 1,024 ports for specific protocols. Port number 21 is for FTP, 23 is for Telnet, 25 is for e-mail, 79 is for finger, 80 is for HTTP, 119 is for Netnews-and the list goes on. It is up to each protocol to determine how a client should interact with the port.

Java and the Net

Java supports TCP/IP both by extending the already established stream I/O interface. Java supports both the TCP and UDP protocol families. TCP is used for reliable stream-based I/O across the network. UDP supports a simpler, hence faster, point-to-point datagram-oriented model.

TCP/IP Client Sockets

TCP/IP sockets are used to implement reliable, bidirectional, persistent, point-to-point, stream-based connections between hosts on the Internet. A socket can be used to connect Java’s I/O system to other programs that may reside either on the local machine or on any other machine on the Internet.

There are two kinds of TCP sockets in Java. One is for servers, and the other is for clients. The `ServerSocket` class is designed to be a “listener,” which waits for clients to connect before doing anything. The `Socket` class is designed to connect to server sockets and initiate protocol exchanges.

The creation of a `Socket` object implicitly establishes a connection between the client and server. There are no methods or constructors that explicitly expose the details of establishing that connection. Here are two constructors used to create client sockets:

`Socket(String hostName, int port)` - Creates a socket connecting the local host to the named host and port; can throw an `UnknownHostException` or an `IOException`.

`Socket(InetAddress ipAddress, int port)` - Creates a socket using a preexisting `InetAddress` object and a port; can throw an `IOException`.

A socket can be examined at any time for the address and port information associated with it, by use of the following methods:

`InetAddress getAddress()` - Returns the `InetAddress` associated with the `Socket` object.

`int getPort()` - Returns the remote port to which this `Socket` object is connected.

`int getLocalPort()` - Returns the local port to which this `Socket` object is connected.

Once the `Socket` object has been created, it can also be examined to gain access to the input and output streams associated with it. Each of these methods can throw an `IOException` if the sockets have been invalidated by a loss of connection on the Net.

1. `InputStream getInputStream()` Returns the `InputStream` associated with the invoking socket.
2. `OutputStream getOutputStream()` Returns the `OutputStream` associated with the invoking socket.

TCP/IP Server Sockets

Java has a different socket class that must be used for creating server applications. The `ServerSocket` class is used to create servers that listen for either local or remote client programs to connect to them on published ports. `ServerSockets` are quite different from normal `Sockets`.

When we create a `ServerSocket`, it will register itself with the system as having an interest in client connections. The constructors for `ServerSocket` reflect the port number that we wish to accept connection on and, optionally, how long we want the queue for said port to be. The queue length tells the system how many client connection it can leave pending before it should simply refuse connections. The default is 50. The constructors might throw an `IOException` under adverse conditions. Here are the constructors:

`ServerSocket(int port)` Creates server socket on the specified port with a queue length of 50.

`ServerSocket(int port, int maxQueue)`-Creates a server socket on the specified port with a maximum queue length of `maxQueue`.

`ServerSocket(int port, int maxQueue, InetAddress localAddress)`-Creates a server socket on the specified port with a maximum queue length of `maxQueue`. On a multihomed host, `localAddress` specifies the IP address to which this socket binds.

`ServerSocket` has a method called `accept()`, which is a blocking call that will wait for a client to initiate communications, and then return with a normal `Socket` that is then used for communication with the client.

JDBC

In an effort to set an independent database standard API for Java, Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMS. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

Swings

Swing is a set of classes that provides more powerful and flexible components than are possible with the AWT. In addition to the familiar components, such as buttons, check boxes, and labels, Swing supplies several exciting additions, including tabbed panes, scroll panes, trees, and tables. Even familiar components such as buttons have more capabilities in Swing. For example, a button may have both an image and a text string associated with it. Also, the image can be changed as the state of the button changes. Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and, therefore, are platform-independent. The term lightweight is used to describe such elements. The number of classes and interfaces in the Swing packages is substantial, and this chapter provides an overview of just a few. Swing is an area that you will want to explore further on your own. The Swing component classes that are used in this book are shown here:

Class	Description
<code>AbstractButton</code>	Abstract superclass for Swing buttons.
<code>ButtonGroup</code>	Encapsulates a mutually exclusive set of buttons.
<code>ImageIcon</code>	Encapsulates an icon.
<code>JApplet</code>	The Swing version of Applet.
<code>JButton</code>	The Swing push button class.
<code>JCheckBox</code>	The Swing check box class.
<code>JComboBox</code>	Encapsulates a combo box
<code>JLabel</code>	The Swing version of a label.

JRadioButton	The Swing version of a radio button.
JScrollPane	Encapsulates a scrollable window.
JTabbedPane	Encapsulates a tabbed window.
JTable	Encapsulates a table-based control.
JTextField	The Swing version of a text field.
JTree	Encapsulates a tree-based control.

Chapter 4

The functioning of the system

The main steps of the processing are: image acquisition and enhancement, plate location and segmentation, character segmentation, character recognition, character validation and registration number validation.

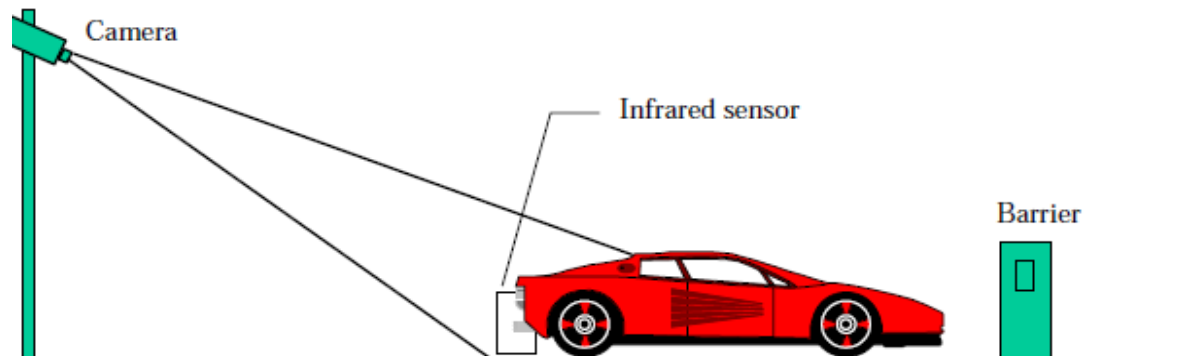


Fig :-The system setup for a parking lot

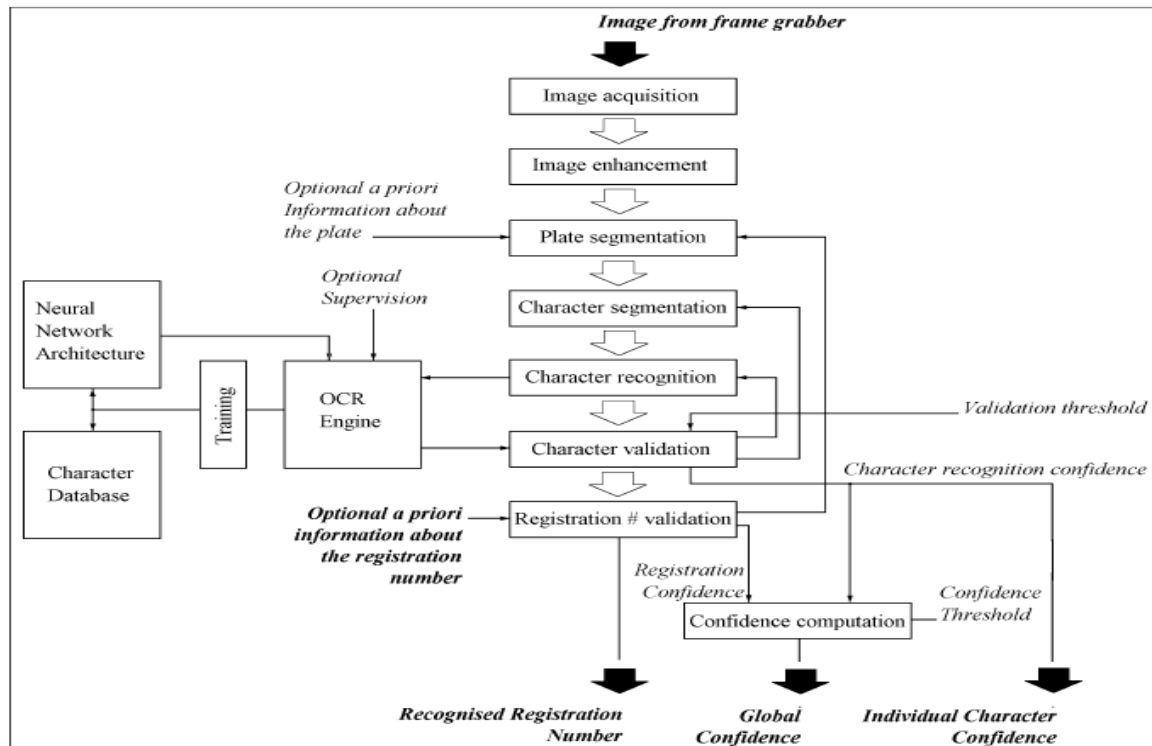


Fig. 2 Some examples of images captured by the system in various external conditions. The first three images (starting from the top left) were captured during daytime. The last three were captured during the night. Picture #2 has a strong back lighting, picture #3 contains a plate with the last character obscured by dirt (7?, Y?, 1?) and picture #6 has a very strong luminance gradient on the horizontal direction.

Plate location

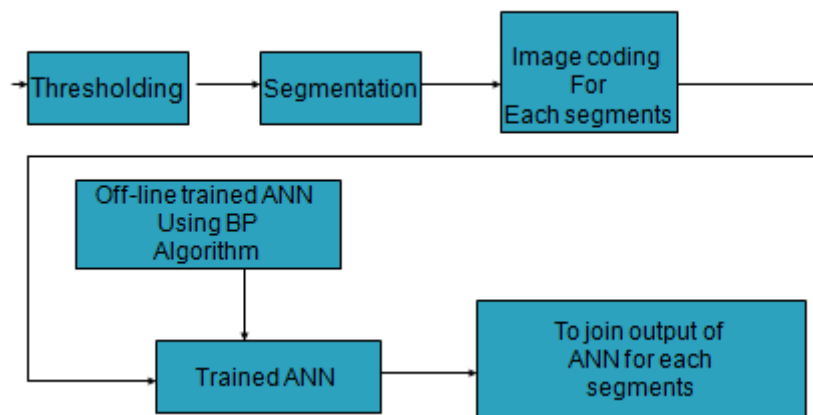
Approach

The approach used for the plate location was to scan the image horizontally looking for repeating contrast changes on a scale of 15 pixels³ and more. This approach uses the assumptions that the contrast between the characters and the background of the plate is sufficiently good, that there are at least 3-4 characters on a plate and that the characters have a minimum vertical size of about 15 pixels (note that this can always be achieved by adjusting the optics of the camera)



The block structure of the system

Architecture Diagram



Processing

A gaussian blur filter is applied to eliminate the fine grain noise. Then, the system calculates the histogram of the image and stretches the histogram with: $\text{newpixel} = \text{pixel} * \gamma + \beta$ where γ and β are calculated so that the stretched histogram will extend on the entire range of grey levels available (from 0 to 255). Subsequently, the program scans the image looking for areas with high contrast gradients at the given scale of about 15 pixels.

The resulting image is scanned again looking for concentrations of such high contrast gradient areas. Any concentration of such areas which can be approximated by a rectangle will be signaled as an interest zone. All subsequent processing will be performed in turn on each interest zone. The first step performed on an interest zone is an image enhancement through another histogram stretching.



Fig. 4 Some processing of 4 interest areas. The first image in each group shows the interest zone in the original image (resampled). The second one shows the same area after histogram analysis and enhancement.

Plate segmentation

Approach

The plate segmentation is performed using a differential gradient edge detection approach. The processing speed is improved by approximating the magnitude of the local edge with the maximum of the absolute values of the gradients on the x and y directions [Abdou, 1979; Foglein, 1983; Davies, 1990]. The reliability is improved by gaussian filtering the edge image and averaging over different edge sensitivities. This approach made the assumptions that the area of the interest zone which actually contains the characters is characterised by a high spatial gradient of intensity (i.e. many character edges). Furthermore, it was assumed that the characters are distributed on one or more rows which are more or less horizontal and the horizontal distribution of the characters is more or less uniform (e.g. one does not have inter character spaces larger than two characters).

Processing

First, the system performs a sobel operation with average preserving templates for vertical and horizontal edges. The result of this operation is binarised with a given threshold and filtered with a gaussian filter. The result is binarised again. Lateral histograms are calculated by projecting vertically and horizontally the resulting binary image. These two lateral histograms will show the number of white pixels (corresponding to edges in the original image) for each vertical and horizontal co-ordinate. The above steps are repeated a given number of times using different thresholds in the binarisation step, and average horizontal and vertical histograms are calculated. These histograms are smoothed with a one dimensional gaussian filtering. The averaging will increase the signal/noise ratio eliminating a lot of noise and preserving only those edges coming from salient features of the image. Usually, the system performs between 9 and 16 sobel-gauss-threshold cycles. Experiments showed that this averaged sobel-gauss thresholding combination is essential and it is able to eliminate many problems like non-uniform illumination, reflections, dirt, etc. The exact number of cycles is decided by the system on an ad-hoc basis depending on the quality of the image and the results obtained.

Character segmentation

Approach

The character segmentation is performed in the following two steps:

Find the number and location of the horizontal group(s) using binarisation and lateral histogram analysis.

For each horizontal group, find the number and location of the characters which form the group using lateral histogram analysis.

The lateral histogram analysis approach was considered the most suitable for this particular application because the edges of the characters can be blurred, noise of various types can partially cover characters or make connections

between different characters and/or characters and borders. In all these situations, classical methods based on morphological analysis of the characters (dilate, erode, closure, skeleton analysis, etc.) and edge detection [Davies, 1990; Marr, 1980] showed to be rather unreliable.

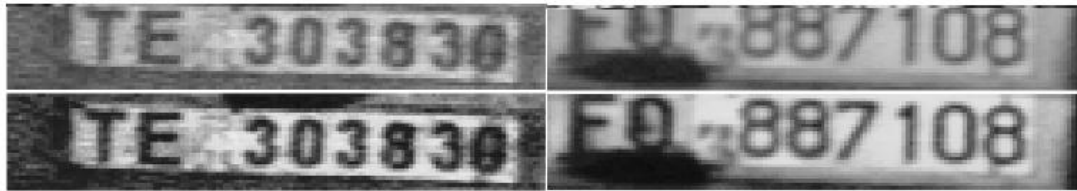


Fig. 5 Examples of plates with partially occluded characters. Note that in the second plate, the dirt connects two characters with the border.

After obtaining this binary image of the interest zone, the system performs the following steps:

1. Find the number and location of the horizontal group(s) by projecting horizontally (one horizontal line is projected to a unique value) the binary image and analysing the resulting lateral histogram.
2. For each horizontal group, find the number and location of the characters by projecting vertically (one vertical line is projected to a unique value) the binarised image of the interest zone.)
3. Any failure will be reported back to the previous module which can re-adjust the binarisation and the plate segmentation.

Chapter 5

Principles of number plate

Area detection

The first step in a process of automatic number plate recognition is a detection of a number plate area. This problematic includes algorithms that are able to detect a rectangular area of the number plate in an original image. Humans define a number plate in a natural language as a “small plastic or metal plate attached to a vehicle for official identification purposes”, but machines do not understand this definition as well as they do not understand what “vehicle”, “road”, or whatever else is. Because of this, there is a need to find an alternative definition of a number plate based on descriptors that will be comprehensible for machines. Let us define the number plate as a “rectangular area with increased occurrence of horizontal and vertical edges”. The high density of horizontal and vertical edges on a small area is in many cases caused by contrast characters of a number plate, but not in every case. This process can sometimes detect a wrong area that does not correspond to a number plate. Because of this, we often detect several candidates for the plate by this algorithm, and then we choose the best one by a further heuristic analysis.

Let an input snapshot be defined by a function $f(x, y)$ where x and y are spatial coordinates, and f is an intensity of light at that point. This function is always discrete on digital computers. We define operations such as edge detection or rank filtering as mathematical transformations of function f . The detection of a number plate area consists of a series of convolve operations. Modified snapshot is then projected into axes x and y . These projections are used to determine an area of a number plate.

Convolution matrices

Each image operation (or filter) is defined by a convolution matrix. The convolution matrix defines how the specific pixel is affected by neighboring pixels in the process of convolution. Individual cells in the matrix represent the neighbors related to the pixel situated in the centre of the matrix. The pixel represented by the cell y in the destination image (fig. 2.1) is affected by the pixels $x_0 \dots x_8$ according to the formula:

$$y = x_0 \times m_0 + x_1 \times m_1 + x_2 \times m_2 + x_3 \times m_3 + x_4 \times m_4 + x_5 \times m_5 + x_6 \times m_6 + x_7 \times m_7 + x_8 \times m_8$$

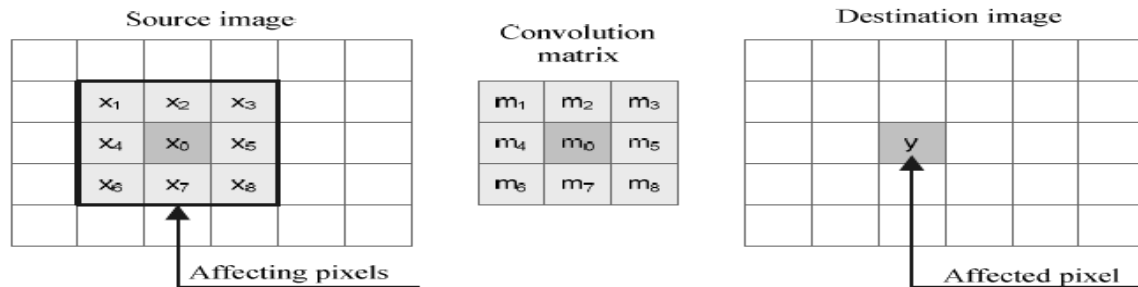


Figure 5.1:-The pixel is affected by its neighbors according to the convolution matrix

Sobel edge detector

The Sobel edge detector uses a pair of 3x3 convolution matrices. The first is dedicated for evaluation of vertical edges, and the second for evaluation of horizontal edges.

$$\mathbf{G}_x = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}; \mathbf{G}_y = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

The magnitude of the affected pixel is then calculated using the formula

$$|G| = \sqrt{G_x^2 + G_y^2}.$$

Horizontal and vertical rank filtering

Horizontally and vertically oriented rank filters are often used to detect clusters of high density of bright edges in the area of the number plate. The width of the horizontally oriented rank filter matrix is much larger than the height of the matrix ($w \gg h$), and vice versa for the vertical rank filter ($w \ll h$). To preserve the global intensity of an image, it is necessary to each pixel be replaced with an average pixel intensity in the area covered by the rank filter matrix. In general, the convolution matrix should meet the following condition:

$$\sum_{i=0}^{w-1} \sum_{j=0}^{h-1} m_{hr}[i, j] = 1.0$$

Where w and h are dimensions of the matrix.

The following pictures show the results of application of the rank and edge detection filters



Figure 5.2:-(a) Original image (b) Horizontal rank filter (c) Vertical rank filter (d)Sobel edge detection (e) Horizontal edge detection (f) Vertical edge detection

Horizontal and vertical image projection

After the series of convolution operations, we can detect an area of the number plate according to a statistics of the snapshot. There are various methods of statistical analysis. One of them is a horizontal and vertical projection of an image into the axes x and y .

The vertical projection of the image is a graph, which represents an overall magnitude of the image according to the axis y (see figure 2.3). If we compute the vertical projection of the image after the application of the vertical edge detection filter, the magnitude of certain point represents the occurrence of vertical edges at that point. Then, the vertical projection of so transformed image can be used for a vertical localization of the number plate. The horizontal projection represents an overall magnitude of the image mapped to the axis x .

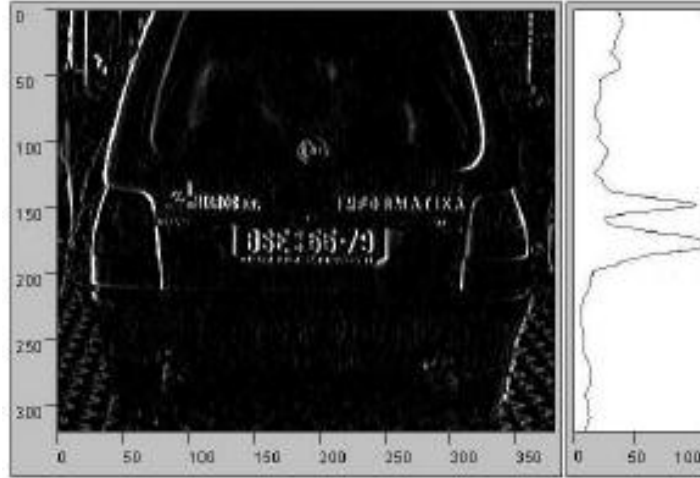


Figure 2.3: Vertical projection of image to a y axis

Let an input image be defined by a discrete function $f(x, y)$. Then, a vertical projection p_y of the function f at a point y is a summary of all pixel magnitudes in the y th row of the input image. Similarly, a horizontal projection at a point x of that function is a summary of all magnitudes in the x th column.

We can mathematically define the horizontal and vertical projection as:

$$p_x(x) = \sum_{j=0}^{h-1} f(x, j) \quad ; \quad p_y(y) = \sum_{i=0}^{w-1} f(i, y)$$

where w and h are dimensions of the image.

Snapshot

Assume the snapshot is represented by a function $f(x, y)$, where $x_0 \leq x \leq x_1$ and $y_0 \leq y \leq y_1$.

The (x_0, y_0) represents the upper left corner of the snapshot, and (x_1, y_1) represents the bottom right corner. If w and h are dimensions of the snapshot, then $x_0 \geq 0$, $y_0 \geq 0$, $x_1 \leq w-1$ and $y_1 \leq h-1$.

Plate

Similarly, the plate p in the band b is an arbitrary rectangle $(x_{p0}, y_{p0}, x_{p1}, y_{p1})$, such as:

$$(x_{b0} \leq x_{p0} \leq x_{p1} \leq x_{b1}) \wedge (y_{p0} = y_{b0}) \wedge (y_{p1} = y_{b0})$$

The band can be also defined as a vertical selection of the snapshot, and the plate as a horizontal selection of the band. The figure 5.4 schematically demonstrates this concept

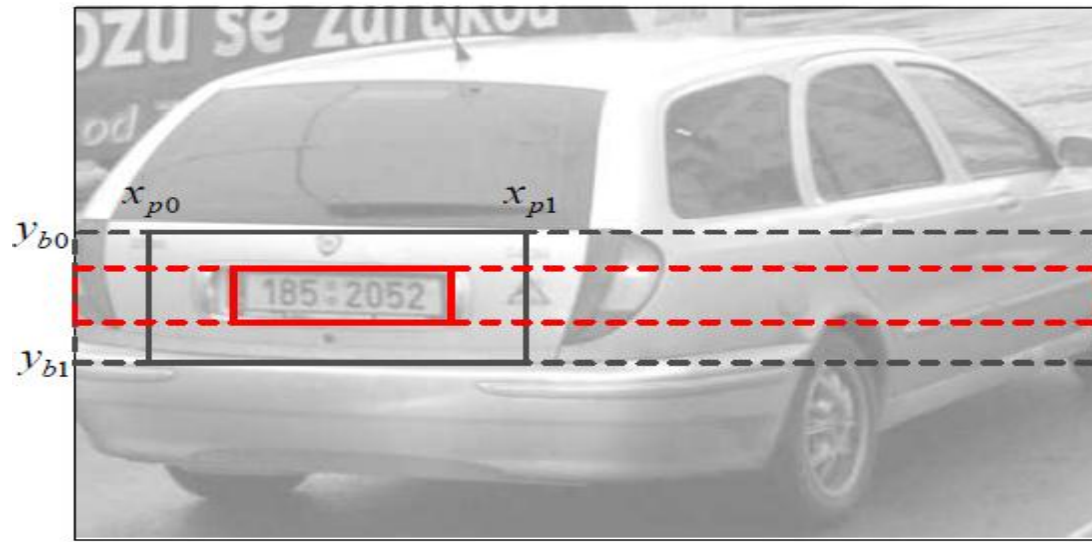


Figure 5.4:-The double-phase plate clipping. Black color represents the first phase of plate clipping, and red color represents the second one. Bands are represented by dashed lines, and plates by solid lines.

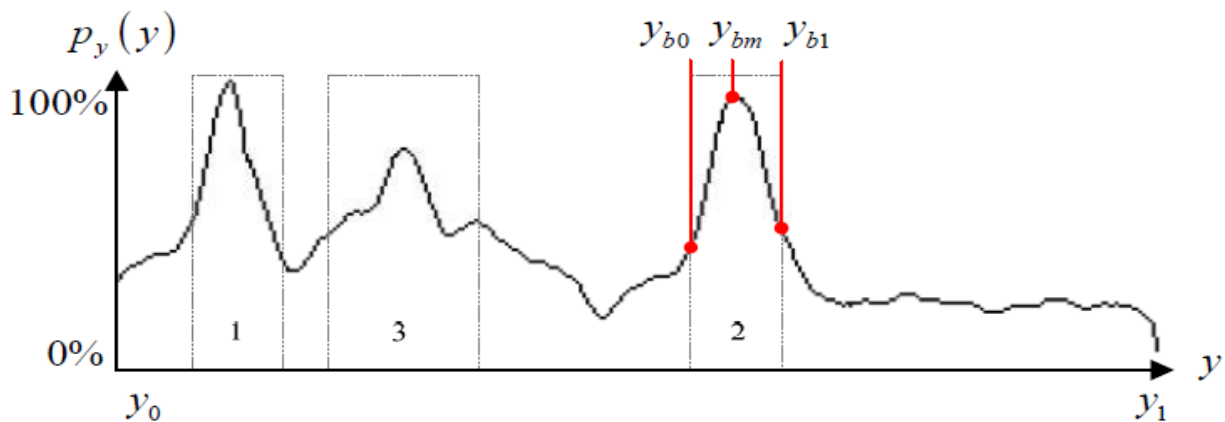


Figure 5.5:-The vertical projection of the snapshot 2.3 after convolution with a rank vector. The figure contains three detected candidates. Each highlighted area corresponds to one detected band.

The fundamental problem of analysis is to compute peaks in the graph of vertical projection.

This principle is applied iteratively to detect several possible bands. The y_{b0} and y_{b1} coordinates are computed in each step of iterative process. After the detection, values of projection p_y in interval are zeroized.

Horizontal detection – plate clipping

In contrast with the band clipping, there is a difference between the first and second phase of plate clipping.

First phase

There is a strong analogy in a principle between the band and plate clipping. The plate clipping is based on a horizontal projection of band. At first, the band must be processed by a vertical detection filter. If w is a width of the band (or a width of the analyzed image), the corresponding horizontal projection $p_x^r(x)$ contains w values:

$$p_x(x) = \sum_{j=y_{b0}}^{y_{b1}} f(x, j)$$

Second phase

In the second phase of detection, the horizontal position of a number plate is detected in another way. Due to the skew correction between the first and second phase of analysis, the wider plate area must be duplicated into a new bitmap. Let $\square f_n(x, y)$ be a corresponding function of such bitmap. This picture has a new coordinate system, such as $[0, 0]$ represents the upper left corner and $[w-1, h-1]$ the bottom right, where w and h are dimensions of the area. The wider area of the number plate after deskewing is illustrated in figure 2.8.

In contrast with the first phase of detection, the source plate has not been processed by the vertical detection filter. If we assume that plate is white with black borders, we can detect that borders as black-to-white and white-to-black transitions in the plate. The horizontal projection $p_x(x)$ of the image is illustrated in the figure 2.7.a. To detect the black-to-white and white-to-black transitions, there is a need to compute a derivative of the projection $p_x(x)$. Since the projection is not continuous, the derivation step cannot be an infinitely small number.

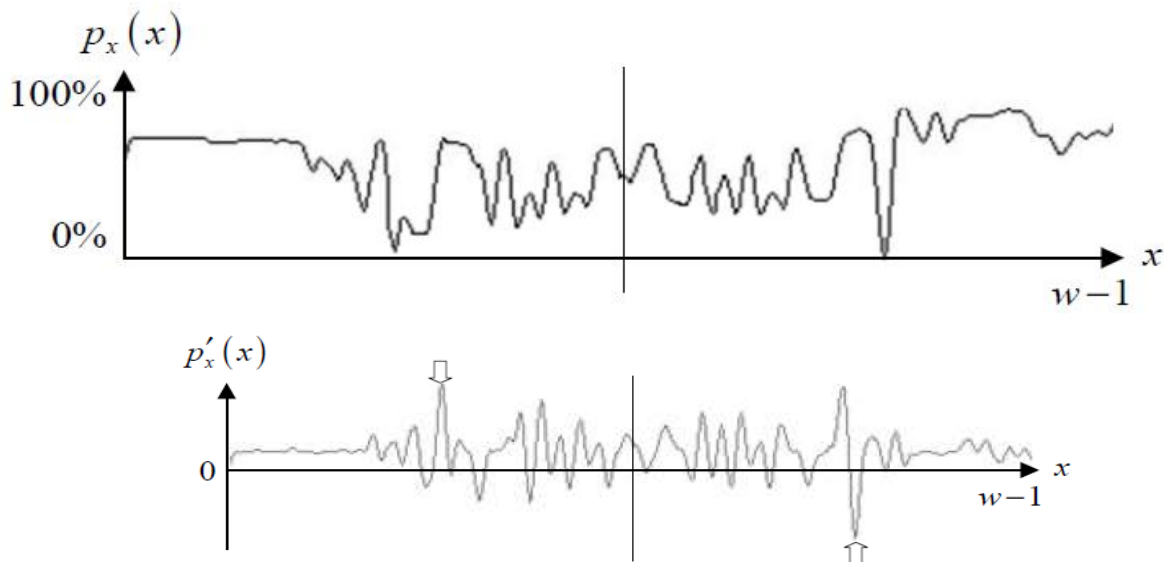


Figure 2.7: (a) The horizontal projection $p_x(x)$ of the plate in figure 2.8. (b) The derivative of $p_x(x)$. Arrows denote the “BW” and “WB” transitions, which are used to determine the boundaries of the plate.



Figure 2.8: The wider area of the number plate after deskewing.

Heuristic analysis and priority selection of number plate

Candidates

In general, the captured snapshot can contain several number plate candidates. Because of this, the detection algorithm always clips several bands, and several plates from each band. There is a predefined value of maximum number of candidates, which are detected by analysis of projections. By default, this value is equals to nine.

There are several heuristics, which are used to determine the cost of selected candidates according to their properties.


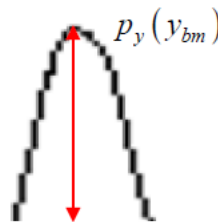
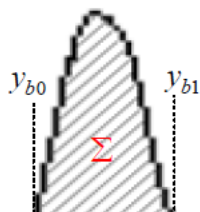
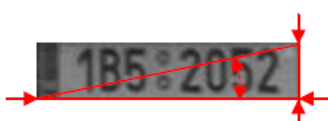
These heuristics have been chosen adhoc during the practical experimentations. The recognition logic sorts candidates according to their cost from the most suitable to the least suitable. Then, the most suitable candidate is examined by a deeper heuristic analysis. The deeper analysis definitely accepts, or rejects the candidate. As there is a need to analyze individual characters, this type of analysis consumes big amount of processor time.

The basic concept of analysis can be illustrated by the following steps:

1. Detect possible number plate candidates.
2. Sort them according to their cost (determined by a basic heuristics).
3. Cut the first plate from the list with the best cost.
4. Segment and analyze it by a deeper analysis (time consuming).
5. If the deeper analysis refuses the plate, return to the step 3.
6. Priority selection and basic heuristic analysis of bands

The basic analysis is used to evaluate the cost of candidates, and to sort them according to this cost. There are several independent heuristics, which can be used to evaluate the cost $I \square \square$. The heuristics can be used separately, or they can be combined together to compute an overall cost of candidate by a weighted sum:

$$\alpha = 0.15 \cdot \alpha_1 + 0.25 \cdot \alpha_2 + 0.4 \cdot \alpha_3 + 0.4 \cdot \alpha_4$$

Heuristics	Illustration	Description
$\alpha_1 = y_{b0} - y_{b1} $		The height of band in pixels. Bands with a lower height will be preferred.
$\alpha_2 = \frac{1}{p_y(y_{bm})}$		The “ $p_y(y_{bm})$ ” is a maximum value of peak of vertical projection of snapshot, which corresponds to the processed band. Bands with a higher amount of vertical edges will be preferred.
$\alpha_3 = \frac{1}{\sum_{y=y_{b0}}^{y_{b1}} p_y(y)}$		This heuristics is similar to the previous one, but it considers not only the value of the greatest peak, but a value of area under the graph between points y_{b0} and y_{b1} . These points define a vertical position of the evaluated band.
$\alpha_4 = \left \frac{ x_{p0} - x_{p1} }{ y_{b0} - y_{b1} } - 5 \right $		The proportions of the one-row number plates are similar in the most countries. If we assume that width/height ratio of the plate is about five, we can compare the measured ratio with the estimated one to evaluate the cost of the number plate.

Deskewing mechanism

The captured rectangular plate can be rotated and skewed in many ways due to the positioning of vehicle towards the camera. Since the skew significantly degrades the recognition abilities, it is important to implement additional mechanisms, which are able to detect and correct skewed plates.

The fundamental problem of this mechanism is to determine an angle, under which the plate is skewed. Then, deskewing of so evaluated plate can be realized by a trivial affine transformation.

It is important to understand the difference between the “sheared” and “rotated” rectangular plate. The number plate is an object in three-dimensional space, which is projected into the two dimensional snapshot during the capture. The positioning of the object can sometimes cause the skew of angles and proportions. If the vertical line of plate v_p is not identical to the vertical line of camera objective v_c , the plate may be sheared. If the vertical lines v_p and v_c are identical, but the axis a_p of plate is not parallel to the axis of camera, a_c the plate may be rotated.

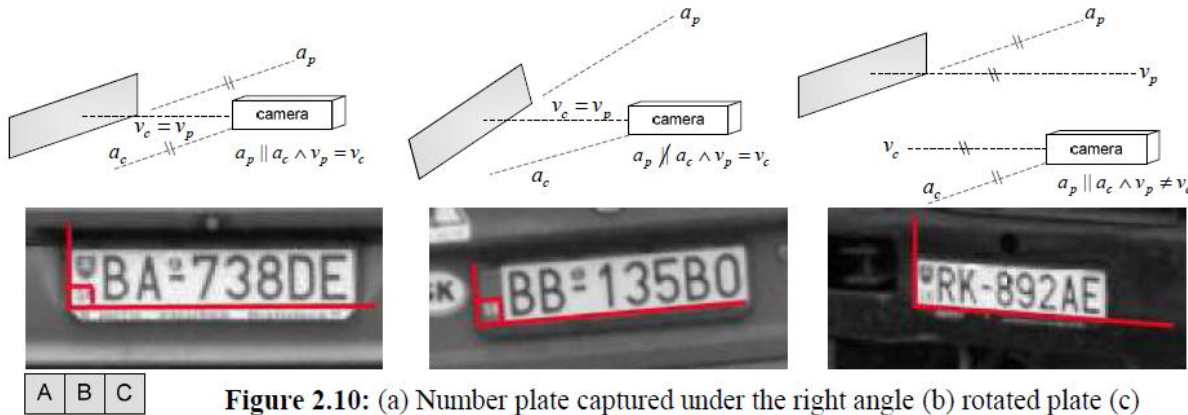


Figure 2.10: (a) Number plate captured under the right angle (b) rotated plate (c) Sheared plate

Detection of skew

Hough transform is a special operation, which is used to extract features of a specific shape within a picture. The classical Hough transform is used for the detection of lines. The Hough transform is widely used for miscellaneous purposes in the problematic of machine vision, but I have used it to detect the skew of captured plate, and also to compute an angle of skew.

It is important to know, that Hough transform does not distinguish between the concepts such as “rotation” and “shear”. The Hough transform can be used only to compute an approximate angle of image in a two-dimensional domain.

The mathematical representation of line in the orthogonal coordinate system is an equation $y = a \cdot x + b$, where a is a slope and b is a y-axis section of so defined line.

Then, the line is a set of all points $[x, y]$, for which this equation is valid. We know that the line contains an infinite number of points as well as there are an infinite number of different lines, which can cross a certain point. The relation between these two assertions is a basic idea of the Hough transform.

The equation $y = a \cdot x + b$ can be also written as $b = -x \cdot a + y$, where x and y are parameters. Then, the equation defines a set of all lines (a, b) , which can cross the point $[x, y]$. For each point in the “XY” coordinate system, there is a line in an “AB” coordinate system (so called “Hough space”)

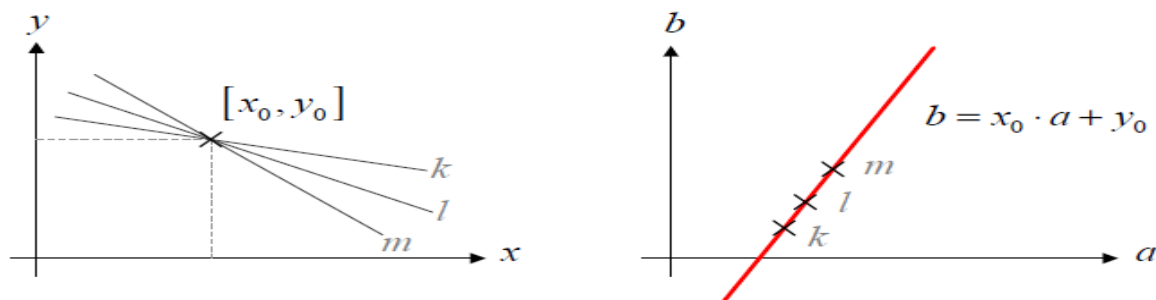


Figure 5.11:-The “XY” and “AB” (“Hough space”) coordinate systems. Each point $[x_0, y_0]$ in the “XY” coordinate system corresponds to one line in the Hough space (red color). The are several points (marked as k, l, m)

in the Hough space, that correspond to the lines in the “XY” coordinate system, which can cross the point (x_0, y_0)

Chapter 6

Principles of plate segmentation

The next step after the detection of the number plate area is a segmentation of the plate. The segmentation is one of the most important processes in the automatic number plate recognition, because all further steps rely on it. If the segmentation fails, a character can be improperly divided into two pieces, or two characters can be improperly merged together. We can use a horizontal projection of a number plate for the segmentation, or one of the more sophisticated methods, such as segmentation using the neural networks. If we assume only one-row plates, the segmentation is a process of finding horizontal boundaries between characters.

The second phase of the segmentation is an enhancement of segments. The segment of a plate contains besides the character also undesirable elements such as dots and stretches as well as redundant space on the sides of character. There is a need to eliminate these elements and extract only the character.

Segmentation of plate using a horizontal projection

Since the segmented plate is deskewed, we can segment it by detecting spaces in its horizontal projection. We often apply the adaptive thresholding filter to enhance an area of the plate before segmentation. The adaptive thresholding is used to separate dark foreground from light background in figure 6.1.a. After the thresholding, we compute a horizontal projection $p_x(x)$ of the plate $f(x, y)$. We use this projection to determine horizontal boundaries between segmented characters. These boundaries correspond to peaks in the graph of the horizontal projection (figure 6.1.b).

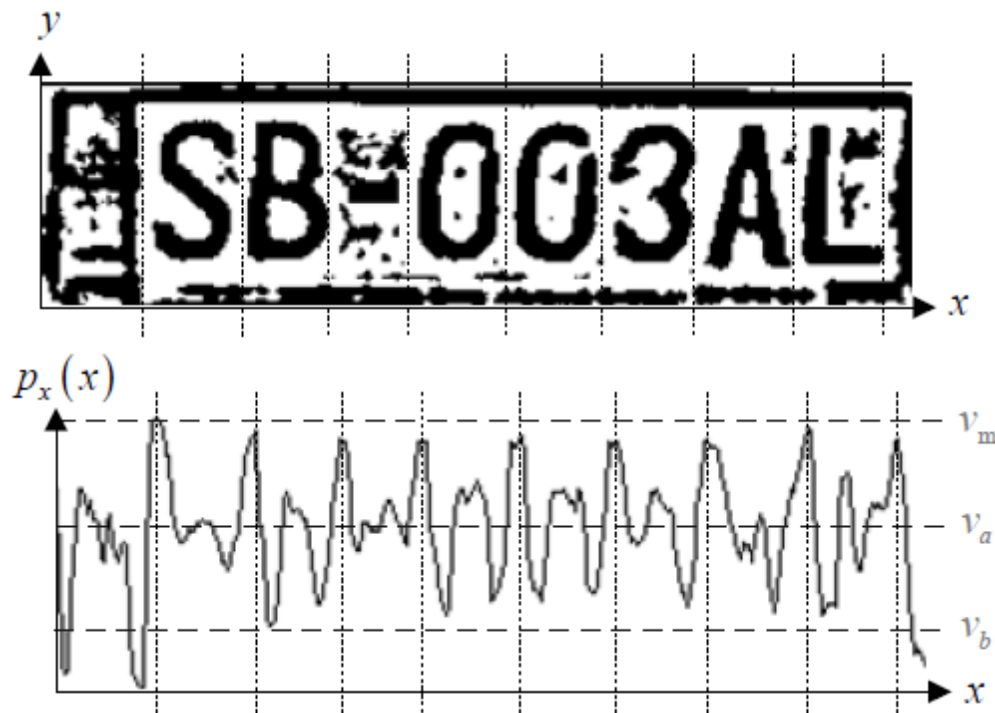


Figure 6.1:-(a) Number plate after application of the adaptive thresholding (b) Horizontal projection of plate with detected peaks. Detected peaks are denoted by dotted vertical lines. The goal of the segmentation algorithm is to find peaks, which correspond to the spaces between characters. At first, there is a need to define several important values in a graph of the horizontal projection $p_x(x)$:

v_m - The maximum value contained in the horizontal projection $p_x(x)$, such as

$$v_m = \max_{0 \leq x < w} \{p_x(x)\}$$

□□□□□□□□□□

□□□□□□

□ v_a - The average value of horizontal projection □ $p_x(x)$ □, such as

$$v_a = \frac{1}{w} \sum_{x=0}^{w-1} p_x(x)$$

□□□□□□□□□□□□□□□□□□

□□□□□□□□□□

□ v_b - This value is used as a base for evaluation of peak height. The base value is always calculated as $v_b = 2$. $v_a - v_m$. The v must lie on vertical axis between the values v_b and v_m .

The algorithm then zeroizes the peak and iteratively repeats this process until no further space is found. This principle can be illustrated by the following steps:

1. Determine the index of the maximum value of horizontal projection:
2. Detect the left and right foot of the peak as:
3. Zeroize the horizontal projection □ $p_x(x)$ □, on interval (x_l, x_r) .
4. If □ $p_x(x) < c_w \cdot v_m$ go to step 7.
5. Divide the plate horizontally in the point x_m .
6. Go to step 1.
7. End.

Two different constants have been used in the algorithm above. The constant c_x is used to determine foots of peak x_m . The optimal value of c_x is 0.7.

The constant c_w determines the minimum height of the peak related to the maximum value of the projection (v_m). If the height of the peak is below this minimum, the peak will not be considered as a space between characters. It is important to choose a value of constant c_w carefully. An inadequate small value causes that too many peaks will be treated as spaces, and characters will be improperly divided. A big value of c_w causes that not all regular peaks will be treated as spaces, and characters will be improperly merged together. The optimal value of c_w is 0.86.

Extraction of characters from horizontal segments

The segment of plate contains besides the character also redundant space and other undesirable elements. We understand under the term “segment” the part of a number plate determined by a horizontal segmentation algorithm. Since the segment has been processed by an adaptive thresholding filter, it contains only black and white pixels. The neighboring pixels are grouped together into larger pieces, and one of them is a character. Our goal is to divide the segment into the several pieces, and keep only one piece representing the regular character. This concept is illustrated in figure 6.2.

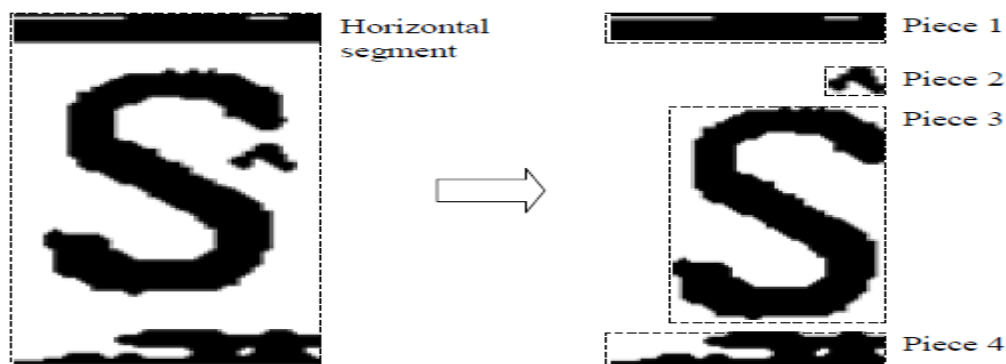


Figure 6.2:-Horizontal segment of the number plate contains several groups (pieces) of neighboring pixels

Algorithm

The goal of the piece extraction algorithm is to find and extract pieces from a segment of the plate. This algorithm is based on a similar principle as a commonly known “seed-fill” algorithm

1. Let piece be a set of (neighboring) pixels x, y
2. Let S be a set of all pieces from a processed segment defined by the function $f(x, y)$.
3. Let X be a set of all black pixels: $X = \{x, y \mid f(x, y) = 1\}$
4. Let A be an auxiliary set of pixels

Principle of the algorithm is illustrated by the following pseudo-code:

```

let set  $S = \emptyset$ 
let set  $X = \{[x, y] \mid f(x, y) = 1 \wedge [0, 0] \leq [x, y] < [w, h]\}$ 
while set  $X$  is not empty do
begin
    let set  $P = \emptyset$ 
    let set  $A = \emptyset$ 
    pull one pixel from set  $X$  and insert it into set  $A$ 
    while set  $A$  is not empty do
    begin
        let  $[x, y]$  be a certain pixel from  $A$ 
        pull pixel  $[x, y]$  from a set  $A$ 
        if  $f(x, y) = 1 \wedge [x, y] \notin A \wedge [0, 0] \leq [x, y] < [w, h]$  then
        begin
            pull pixel  $[x, y]$  from set  $A$  and insert it into set  $P$ 
            insert pixels  $[x-1, y], [x+1, y], [x, y-1], [x, y+1]$  into set  $A$ 
        end
    end
    add  $P$  to set  $S$ 
end

```

Heuristic analysis of pieces

The piece is a set of pixels in the local coordinate system of the segment. The segment usually contains several pieces. One of them represents the character and others represent redundant elements, which should be eliminated. The goal of the heuristic analysis is to find a piece, which represents character.

Pieces with a higher value will be preferred. The piece chosen by the heuristics is then converted to a monochrome bitmap image. Each such image corresponds to one horizontal segment. These images are considered as an output of the segmentation phase of the ANPR process

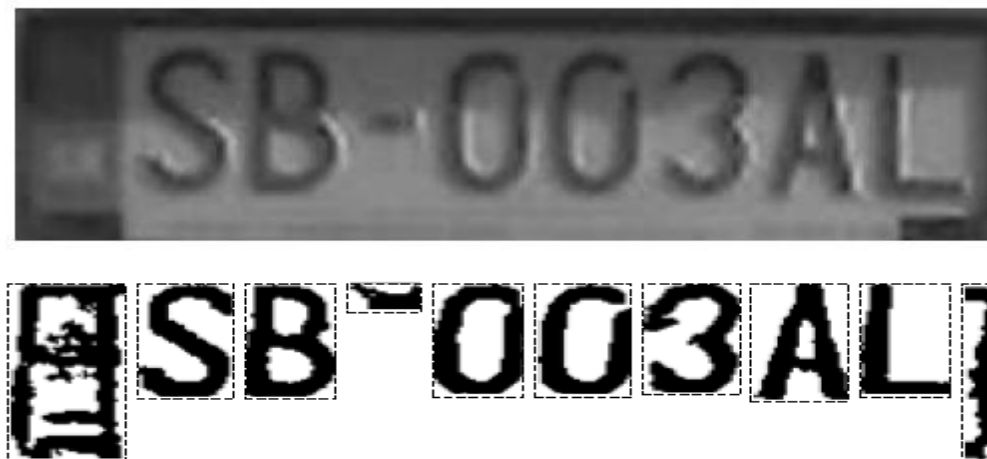


Figure 6.3:-The input (a) and output (b) example of the segmentation phase of the ANPR recognition process.

Chapter 7

Recognition of characters

The previous chapter deals with various methods of feature extraction. The goal of these methods is to obtain a vector of descriptors (so-called pattern), which comprehensively describes the character contained in a processed bitmap. The goal of this chapter is to introduce pattern recognition techniques, such as neural networks, which are able to classify the patterns into the appropriate classes.

General classification problem

The general classification problem is formulated using the mapping between elements in two sets. Let A be a set of all possible combinations of descriptors, and B be a set of all classes. The classification means the projection of group of similar element from the set A into a common class represented by one element in the set B . Thus, one element in the set B corresponds to one class. Usually the group of distinguishable instances of the same character corresponds to the one class, but sometimes one class represents two mutually indistinguishable characters, such as “0” and “O”.

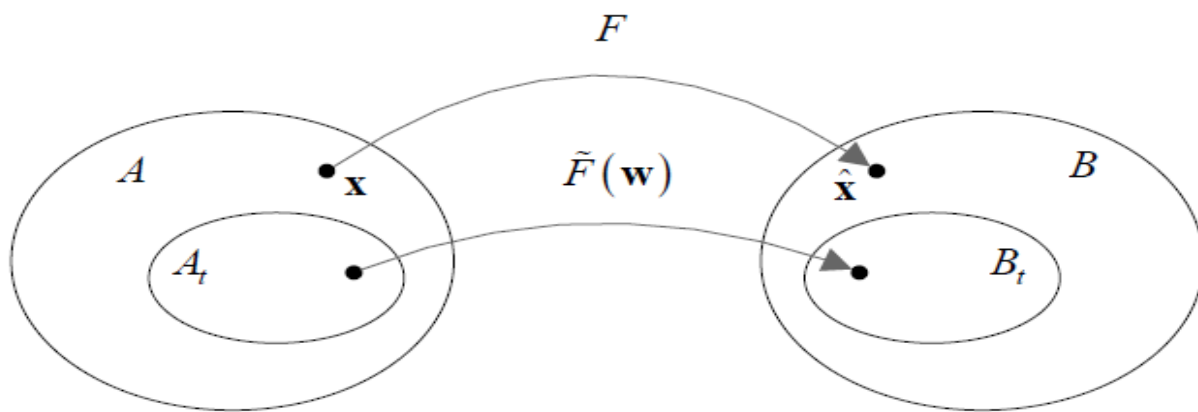


Figure 5.1:-The projection between sets A and B .

Biological neuron and its mathematical models

For a better understanding of artificial neural network architecture, there is a need to explain the structure and functionality of a biological neuron. The human brain is a neural network of about ten billions interconnected neurons. Each neuron is a cell that uses a biochemical reaction to process and transmit information. The neural cell has a body of size about several micrometers and thousands of input connections called “dendrites”. It also has one output connection called “axon”, which can be several meters long. The data flow in the biological neural network is represented by electrical signal, which propagates along the axon. When the signal reaches a synaptic connection between the axon and a consecutive dendrite, it releases molecules of chemical agent (called mediators or neuro-transmitters) into such dendrite. This action causes a local change of polarity of a dendrite transmission membrane. The difference in the polarity of the transmission membrane activates a dendrite-somatic potential wave, which advances in a system of branched dendrites into the body of neuron.

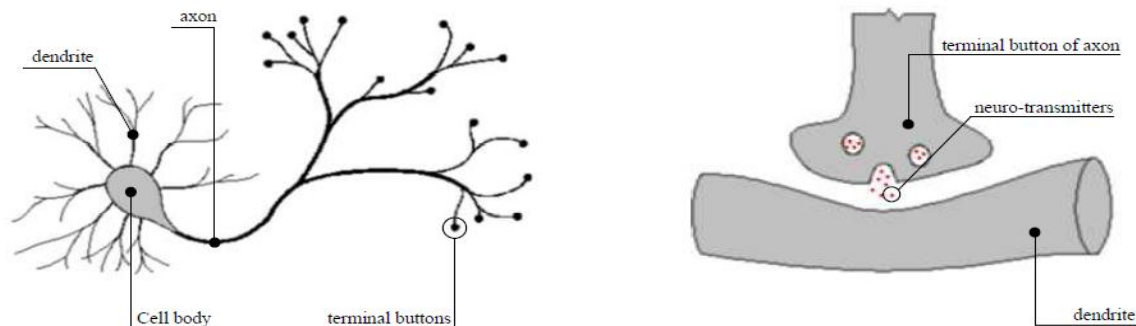


Figure 5.3:-(a) Schematic illustration of the neural cell (b) The synaptic connection between a dendrite and terminal button of the axon

Since the problematic of the biological neuron is very difficult, the scientists proposed several Mathematical models, such as McCulloch-Pitts binary threshold neuron, or the perceptron.

McCulloch-Pitts binary threshold neuron

The McCulloch-Pitts binary threshold neuron was the first model proposed by McCulloch and Pitts in 1943. The neuron has only two possible output values (0 or 1) and only two types of the synaptic weights: the fully excitative and the fully inhibitive. The excitative weight (1) does not affect the input, but the inhibitive one negates it (-1).

This type of neuron can perform logical functions such as AND, OR, or NOT. In addition, McCulloch and Pitts proved that synchronous array of such neurons is able to realize arbitrary computational function, similarly as a Turing machine. Since the biological neurons have not binary response (but continuous), this model of neuron is not suitable for its approximation.

Perceptron

Another model of neuron is a Perceptron. It has been proved that McCulloch-Pitts networks with modified synaptic connections can be trained for the recognition and classification. The training is based on a modification of a neuron weights, according to the reaction of such neuron as follows. If the neuron is not active and it should be, we increase the weights. If the neuron is active and it should not be, we decrease them. This principle was been used in a first model of the neural classifier called ADALINE (adaptive linear neuron). The major problem of such networks is that they are not able to solve linearly non-separable problems. This problem has been solved when the scientists Rumelhart, Hilton and Williams proposed the error back-propagation method of learning for multilayered Perceptron networks. The simple McCulloch-Pitts binary threshold neurons have been replaced by neurons with continuous saturation input/output function. Perceptron has multiple analogous inputs and one analogous output.

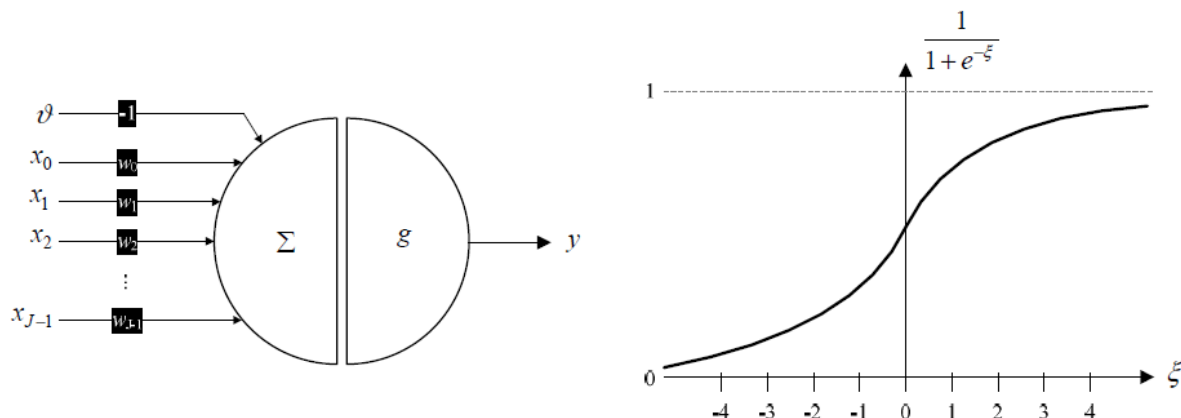


Figure 5.4:-(a) The summation Σ and gain (saturation) g function of the Perceptron with a Threshold implemented as a dedicated input. (b) The sigmoid saturation function.

Feed-forward neural network

Formally, the neural network is defined as an oriented graph $G(N, E)$, where N is a nonempty set of neurons, and E is a set of oriented connections between neurons.

$$N = N_0 \cup N_1 \cup N_2$$

The number of neurons in the input layer (m) is equal to a length of an input pattern x in order that each value of the pattern is dedicated to one neuron. Neurons in the input layer do not perform any computation function, but they only distribute values of an input pattern to neurons in the hidden layer. Because of this, the input layer neuron has only one input directly mapped into multiple outputs. Because of this, the threshold value of the input layer neuron is equal to zero, and the weights of inputs w are equal to one. The number of neurons in the hidden layer (n) is scalable, but it affects the recognition abilities of a neural network at a whole. Too few neurons in the hidden layer causes that the neural network would not be able to learn new patterns. Too many neurons cause network to be over learned, so it will not be able to generalize unknown patterns as well. The information in a feed-forward neural network is propagated from lower layers to upper layers by one-way connections. There are connections only between adjacent layers, thus feed forward neural network does not contain feedback connections, or connections between arbitrary two layers. In addition, there exist no connections between neurons from the same layer.

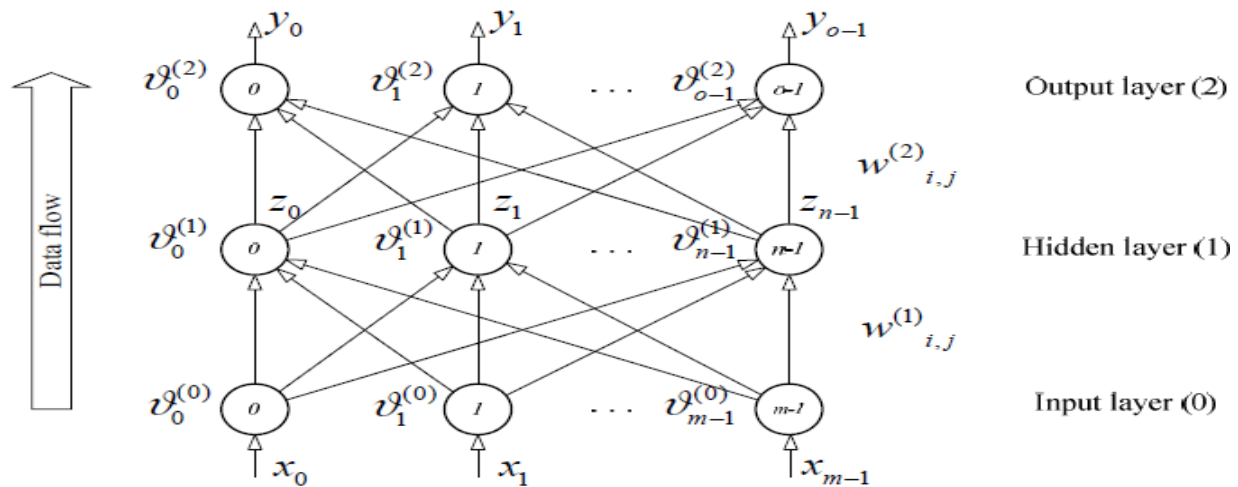


Figure 5.5:-Architecture of the three layer feed-forward neural network.

The error function $t E$ goes down as a number of neurons in the hidden layer grow. This relation is valid also between the function $t E$ and a number of iterative steps of the adaptation process. These relations can be mathematically described as follows:

$$\lim_{n \rightarrow \infty} E_t = 0 ; \lim_{k \rightarrow \infty} E_x = 0$$

Where n is the number of neurons in the input layer and k is the number of iteration steps of the adaptation process. The error function $x E$ does not have a limit at zero as n and k goes to infinity. Because of this, there exists an optimal number of neurons and optimal number of iteration steps, in which the function $x E$ has a minimum.

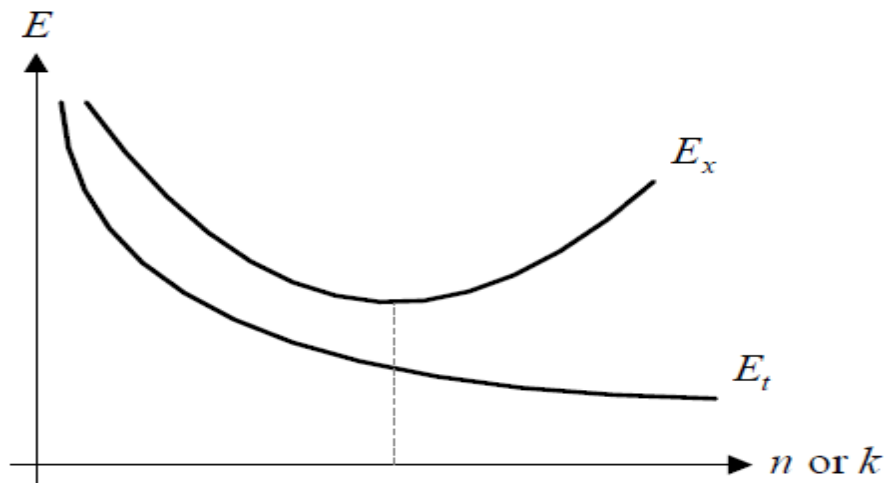


Figure 5.6:-Dependency of error functions $t E$ and $x E$ on the number of neurons in input layer (n) and the number of iteration steps (k).

For simplicity, we will assume only a feed-forward neural network with one layer of hidden neurons defined in section 5.3. All neurons in adjacent layers are connected by oriented connections. There are no feedback connections, or connections between neurons within a single layer.

The activities of hidden and output neurons are defined as:

$$z_i = g \left(\sum_{j=0}^{m-1} w_{i,j}^{(1)} \cdot x_j - \vartheta_i^{(1)} \right) ; y_i = g \left(\sum_{j=0}^{n-1} w_{i,j}^{(2)} \cdot z_j - \vartheta_i^{(2)} \right)$$

activities of neurons in the hidden layer activities of neurons in the output layer

Active phase

Evaluation of the activities of hidden and output neurons is performed in so-called “active phase”. The active phase consists of two steps in three-layer neural networks. The first step is an evaluation of activities z_i in the hidden layer, and the second step is an evaluation of activities. Since the evaluation of activities is performed from bottom layers to top ones, the term “feed-forward” refers to this principle. The active phase corresponds to an approximation $F(x, w)$ of function $F(x)$, and it is performed every time when there is a need to classify the input pattern x . The following pseudo-code demonstrates the active phase of feed-forward neural network.

```

procedure activePhase (input:  w , // vector of thresholds and weights
                        x ; // input pattern to be classified
                        output: z , // vector of activities of neurons in hidden layer
                              y // vector of activities of neurons in output layer (neural
                                network response)
)
begin
    // first step: evaluate activities of neurons in the hidden layer
    for each neuron in hidden layer with index  $i \in 0, \dots, n-1$  do
    begin
        let  $\xi = \mathbf{w}[\vartheta_i^{(1)}]$ 
        for each input with index  $j \in 0, \dots, m-1$  do
            let  $\xi = \xi + \mathbf{w}[w_{i,j}^{(1)}] \cdot x_j$ 
        let  $z_i = g(\xi)$ 
    end

    // second step: evaluate activities of neurons in the output layer
    for each neuron in output layer with index  $i \in 0, \dots, o-1$  do
    begin
        let  $\xi = \mathbf{w}[\vartheta_i^{(2)}]$ 
        for each input with index  $j \in 0, \dots, n-1$  do
            let  $\xi = \xi + \mathbf{w}[w_{i,j}^{(2)}] \cdot z_j$ 
        let  $y_i = g(\xi)$ 
    end
end
end

```

Partial derivatives and gradient of error function

The goal of the training phase is to find optimal values of thresholds and weights to minimize the error function E_t . Adaptation phase is an iterative process in which a response y to an input pattern x is compared with the desired response \hat{x} . The difference between the obtained and desired response is used for a correction of weights. The weights are iteratively altered until the value of the error function E_t is negligible.

Heuristic analysis of characters

The segmentation algorithm described in chapter three can sometimes detect redundant elements, which do not correspond to proper characters. The shape of these elements after normalization is often similar to the shape of characters. Because of this, these elements are not reliably separable by traditional OCR methods, although they

vary in size as well as in contrast, brightness or hue. Since the feature extraction methods described in chapter four do not consider these properties, there is a need to use additional heuristic analyses to filter non-character elements. The analysis expects all elements to have similar properties. Elements with considerably different properties are treated as invalid and excluded from the recognition process. The analysis consists of two phases. The first phase deals with statistics of brightness and contrast of segmented characters. Characters are then normalized and processed by the piece extraction algorithm. Since the piece extraction and normalization of brightness disturbs statistical properties of segmented characters, it is necessary to proceed the first phase of analysis before the application of the piece extraction algorithm. In addition, the heights of detected segments are same for all characters. Because of this, there is a need to precede the analysis of dimensions after application of the piece extraction algorithm. The piece extraction algorithm strips off white padding, which surrounds the character.

Respecting the constraints above, the sequence of steps can be assembled as follows:

1. Segment the plate.
2. Analyze the brightness and contrast of segments and exclude faulty ones.
3. Apply the piece extraction algorithm on segments.
4. Analyze the dimensions of segments and exclude faulty ones.

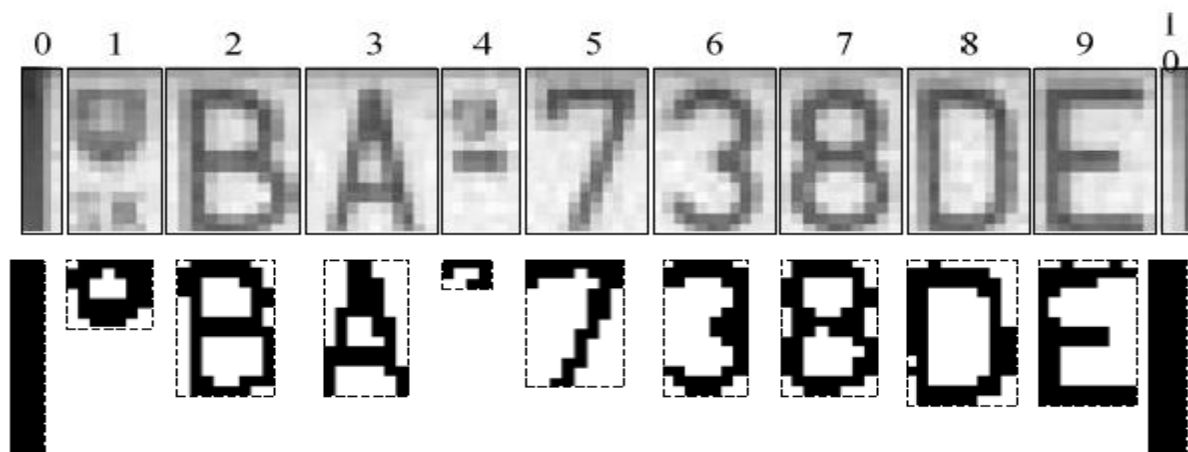


Figure 5.8:-Character segments before (a) and after (b) application of the piece extraction algorithm. This algorithm disturbs statistical properties of brightness and contrast.

If we assume that there are not big differences in brightness and contrast of segments, we can exclude the segments which considerably differ from the mean. Let i th segment of plate be defined by a discrete function, where w_i and h_i are dimensions of the element. We define the following statistical properties of an element

	i	BRI	CON	HUE	SAT	HEI	WHR	Violated constraints
	0	0.247	0.038	0.152	0.236	0.189	0.093	BRI,HUE,WHR
	1	0.034	0.096	0.181	0.134	-0.554	0.833	HUE,HEI
	2	0.002	0.018	0.030	0.038	0.040	0.642	
	3	0.084	0.012	0.003	0.061	0.189	0.625	
	4	0.001	0.003	0.021	0.059	-0.777	1.666	HEI,WHR
	5	0.117	0.016	0.002	0.063	0.189	0.625	
	6	0.063	0.016	0.007	0.056	0.189	0.562	
	7	0.025	0.011	0.025	0.028	0.114	0.533	
	8	0.019	0.025	0.012	0.034	0.114	0.600	
	9	0.019	0.048	0.009	0.045	0.114	0.533	
	10	0.062	0.009	0.041	0.018	0.189	0.095	WHR

Table 5.1:-Properties of segments in figure 5.8. The meaning of abbreviations is as follows:

BRI=brightness, CON=contrast, HUE=hue, SAT=saturation, HEI=height, WHR=width/height ratio.

Chapter 8

Syntactical analysis of

Recognized plate

principle and algorithms

In some situations when the recognition mechanism fails, there is a possibility to detect a failure by a syntactical analysis of the recognized plate. If we have country-specific rules for the plate, we can evaluate the validity of that plate towards these rules. Automatic syntax-based correction of plate numbers can increase recognition abilities of the whole ANPR system. For example, if the recognition software is confused between characters “8” and “B”, the final decision can be made according to the syntactical pattern. If the pattern allows only digits for that position, the character “8” will be used rather than the character “B”. Another good example is a decision between the digit “0” and the character “O”. The very small difference between these characters makes their recognition extremely difficult, in many cases impossible.

Recognized character and its cost

In most cases, characters are recognized by neural networks. Each neuron in an output layer of a neural network typically represents one character. Let $y = [y_0 \dots y_9, y_a \dots y_z]$ be a vector of output activities. If there are 36 characters in the alphabet, the vector y will be also 36-dimensional. Let y_i be an i th component of the vector y . Then, y_i means how much does the input character corresponds to the i th character in the alphabet, which is represented by this component. The recognized character χ is represented by the greatest component of the vector y :

$$\chi = chr \left(\max_{0 \leq i \leq z} \{y_i\} \right)$$

Where $chr(y_i)$ is the character, which is represented by the i th component of vector y . Let $y^{(s)}$ be a vector y descendingly sorted according to the values of components. Then, the recognized character is represented by the first component of so sorted vector:

$$\chi = chr \left(y_0^{(s)} \right)$$

When the recognition process fails, the first component of $y^{(s)}$ can contain invalid character, which does not match the syntax pattern. Then, it is necessary to use the next valid character with a worse cost.

Syntactical patterns

In practice, ANPR systems must deal with many different types of plate numbers. Number plates are not unified, so each country has own type. Because of this, number plate recognition system should be able to recognize a type of the number plate, and automatically assign the correct syntactical pattern to it. The assignation of the right syntactical pattern is a fundamental problem in syntactical analysis. Syntactical pattern is a set of rules defining characters, which can be used on a certain position in a plate number. If the plate number P is a sequence of n alphanumeric characters $P = [p^0 \dots p^{n-1}]$, then the syntactical pattern \mathbf{P} is a n -tuple of sets p^i is a set of all allowed characters for the i th position in a plate. For example, czech number plates can contain digit on a first position followed by a character denoting the region, where the plate has been registered and five other digits for a registration number of a car. Formally, the syntactical pattern \mathbf{P} for number plates can look like this:

$$\mathbf{P} = \left(\begin{array}{l} \{0,1,2,3,4,5,6,7,8,9\}, \{C,B,K,H,L,T,N,E,P,A,S,U,J,Z\}, \\ \{0,1,2,3,4,5,6,7,8,9\}, \{0,1,2,3,4,5,6,7,8,9\}, \{0,1,2,3,4,5,6,7,8,9\}, \\ \{0,1,2,3,4,5,6,7,8,9\}, \{0,1,2,3,4,5,6,7,8,9\}, \{0,1,2,3,4,5,6,7,8,9\} \end{array} \right)$$

Choosing the right pattern

For example, assume that plate number '0B01234' has been recognized as '0801234', and the recognition pattern does not allow digit at the second position of a plate. If the character "8" has been recognized with similarity ratio of 0.90, and other characters with the ratio of 0.95, the metrics for this pattern is determined as follows.

$$\delta(\mathbf{P}) = (1) + \left(\frac{10^{-2}}{0.95} + \frac{10^{-2}}{0.90} + \frac{10^{-2}}{0.95} + \frac{10^{-2}}{0.95} + \frac{10^{-2}}{0.95} + \frac{10^{-2}}{0.95} + \frac{10^{-2}}{0.95} \right) = 1,07426$$

If there is a pattern that exactly matches to the evaluated plate number, we can say that number has been correctly recognized, and no further corrections are needed. In addition, it is not possible to detect a faulty number plate, if it does not break rules of a syntactical pattern. Otherwise, it is necessary to correct detected plate using the pattern with lowest cost □□:

The correction of a plate means the replacement of each invalid character by another one. If the character $p(i)$ at the i th position of the plate P does not match the selected pattern P , it will be replaced by the first valid one from $y(s)$. $y(s)$ is a sorted vector of output activities denoting how much the recognized character is similar to an individual character from the alphabet. Heuristic analysis of a segmented plate can sometimes incorrectly evaluate non-character elements as characters. Acceptance of the non-character elements causes that the recognized plate will contain redundant characters. Redundant characters occur usually on sides of the plate, but rarely in the middle.

Chapter 9

Tests and final

Considerations

Choosing the representative set of snapshots

Many of static snapshots of vehicles for the test purposes. Random moving and standing vehicles with Slovak and Czech number plates have been included. At first, my objective was to find a representative set of number plates, which are recognizable by humans. Of course, the set like this contains extremely wide spectrum of plates, such as clear and easy recognizable as well as plates degraded by the significant motion blur or skew. Then, a recognition ability of a machine is represented by a ratio between the number of plates, which have been recognized by the machine, and the number of plates recognized by a human. Practically, it is impossible to build a machine with the same recognition abilities as a human has. Because of this, the test like this is extremely difficult and useless. In praxis, it is more useful to find a representative set of number plates, which can be captured by an ANPR camera. The position of the camera has a significantly affects the quality of captured images, and a successfulness of the whole recognition process. The suitable position of the camera towards the lane can lead to a better set of all possible snapshots. In some situations, we can avoid of getting skewed snapshots by a suitable positioning of the camera. Sometimes, this is cleverer than a development of the robust de-skewing mechanisms.

Let S be a representative set of all snapshots, which can be captured by a concrete instance of the ANPR camera. Some of the snapshots in this set can be blurred; some of them can be too small, too big, too skewed or too deformed.

$$S = S_c \cup S_b \cup S_s \cup S_e \cup S_l$$

Where S^c a subset of "clear" plates is, S^b is a subset of blurred plates, S is a subset of skewed plates, which has a difficult surrounding environment, and l S is a subset of plates with little characters.



Evaluation of a plate number correctness

Plate numbers recognized by a machine can sometimes differ from the correct ones. Because of this, there is a need to define formulas and rules, which will be used to evaluate a degree of plate correctness. Let P be a plate number, and $S = \{P(0), \dots, P(n-1)\}$ be a set of all tested plate numbers. Then, recognition rate $R(S)$ of the ANPR system tested on set S is calculated as:

$$R(S) = \frac{1}{n} \sum_{i=0}^{n-1} s(P^{(i)}),$$

where n is a cardinality of the set S , and $s(P)$ is a correctness score of the plate P . The correctness score is a value, which express how successfully the plate has been recognized. Now the question is how to define the correctness score of individual plates. There are two different approaches, how to evaluate it. The first is a binary score, and the second is a weighted score.

Binary score

Let us say, that plate number P is a sequence of n alphanumerical characters $P = \{p(0), \dots, p(n-1)\}$. If $P(r)$ is the plate number recognized by a machine, and $P(c)$ is the correct one, then binary score s_b of plate $P(r)$ is evaluated as follows:

$$s_b(P^{(r)}) = \begin{cases} 0 & \text{if } P^{(r)} \neq P^{(c)} \\ 1 & \text{if } P^{(r)} = P^{(c)} \end{cases}$$

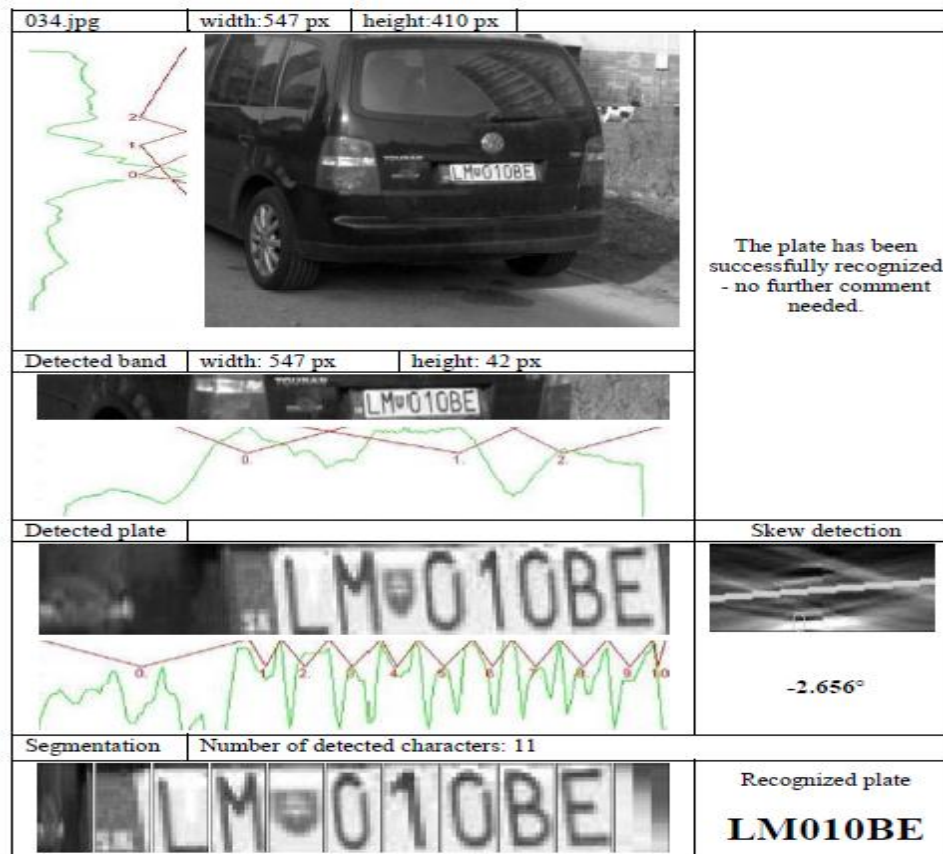
Results:-

The table 7.1 shows recognition rates, which has been achieved while testing on various set of number plates. According to the results, this system gives good responses only to clear plates, because skewed plates and plates with difficult surrounding environment causes significant degradation of recognition abilities.

	Total number of plates	Total number of characters	Weighted Score
Clear plates	68	470	87.2
Blurred plates	52	352	46.87
Skewed plates	40	279	51.64
Average plates	177	1254	73.02

Table 7.1:-Recognition rates of the ANPR system

Snap Shots



APPLICATIONS

Capturing a vehicle license plate using a specialized video camera
 Generating alert notifications as 'target vehicles' are identified (Sounding the Alarm)
 Analysis of city traffic during peak periods
 Flexible and automatic highway toll collection systems
 Enhanced vehicle theft prevention
 Effective enforcement of traffic rule

Advantages

1. Works regardless of lighting and weather conditions.
2. Can be easily integrated into most security systems.
3. Suitable for wide range of locations.

Disadvantages

1. Poor image resolution, usually because the plate is too far away but sometimes resulting from the use of a low-quality camera.
2. Poor lighting and low contrast due to overexposure, reflection or shadows.

Future Enhancements

It is needed another method for plate localization to prevent failures of the plate location.

Representation should be more improved in the future so that each typical character can be known so that the recognition process can be more precise

The implementation can be made more accurate

Conclusion:-

The objective of this thesis was to study and resolve algorithmic and mathematical aspects of the automatic number plate recognition systems, such as problematic of machine vision, pattern recognition, OCR and neural networks.

This work also contains demonstration ANPR software, which comparatively demonstrates all described algorithms.

ANPR solution has been tested on static snapshots of vehicles, which has been divided into several sets according to difficultness. Sets of blurry and skewed snapshots give worse recognition rates than a set of snapshots, which has been captured clearly.

The objective of the tests was not to find a one hundred percent recognizable set of snapshots, but to test the invariance of the algorithms on random snapshots systematically classified to the sets according to their properties.

Bibilography:-

1. Fajmon B.: Numeric Math and Probability, scripts, Faculty of Electrical Engineering and Communication, Brno, Czech Republic, 2005
2. Fraser N.: Introduction to Neural Networks, <http://www.virtualventures.ca/~neil/neural/neuron.html>
3. Fukunaga K.: Introduction to statistical pattern recognition, Academic Press, San Diego, USA, 1990
4. Gonzalez R., Woods R.: Digital Image Processing, Prentice Hall, Upper Saddle River, New Jersey, 2002
5. Kovar M.: Discreet Math, scripts, Faculty of Electrical Engineering and Communication, Brno, Czech Republic, 2003
6. Kuba M.: Neural Networks, scripts, Faculty of Informatics, Masaryk University, Brno, Czech Republic
7. Kvasnicka V., Benuskova L., Pospichal J., Farkas I., Tino P., Kral A.: Introduction to Neural Networks, Technical University, Kosice, Slovak Republic.