LICENSE PLATE RECOGNITION

SYSTEM

By

NC Shadab Ghaffar

Capt Shahzad Khadim

NC Ali Arsalan

PC Muhammad Khalil Ur Rahman

<u>Supervised by</u>

Lt. Col. Naveed Sarfaraz Khattak

Submitted to the Faculty of Computer Science,
Military College of Signals, National University of Sciences and Technology,
Rawalpindi in partial fulfillment of B.E. degree in Computer Software Engineering.

April, 2006

ABSTRACT

LICENSE PLATE RECOGNITION SYSTEM

Many facilities charge per vehicle entry for usage. Such facilities include parking areas, highways, bridges or tunnels. It is common to have staff at the checkpoints to control the payment of fees or to have machines if a set fee is required. The purpose of this research is to investigate a suitable way to read and recognize the registration plate from a photograph of a vehicle. An automated system could then be implemented to control the payment of fees. In fact, any situation requiring the automatic control of identification of a vehicle with a license plate number may be a potential application for such a system. At roads there is a need of a system which can detect stolen cars and also suspicious vehicles for security purposes.

This report describes analysis, design and implementation of a system for automatic recognition of license plates, which is considered a subsystem for automatic speed control. The input to the system is a series of color images of moving vehicles, and output consists of the registration number of the license plate.

Extraction of the desired information is done in three steps. First, the license plate is extracted from the original image, then the characters are isolated, and finally each character is identified using statistical pattern recognition and correlation.

The algorithms were developed using a set of training images, and tested on images taken under varying conditions. The final program is capable of extracting the desired information in a high percentage of the test images.

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another reward or qualification either in this institute or elsewhere.

DEDICATION

In the name of Allah, the Most merciful, the Most Beneficent

To our dear parents and siblings…especially to our Mothers

ACKNOWLEDGMENTS

After the grace of Almighty Allah and the prayers and support of our parents and family, we wish to express our sincere gratitude to our supervisor Lt. Col. Naveed Sarfraz Khattak for his guidance, technical assistance and inspiration throughout the project. We would like to extend our gratitude to Maj. Athar Mohsin Zaidi whose critics and his moral support has leaded the research towards the ultimate goal.

We would like to take the opportunity to honor our fathers and mothers, whose inspiration and support enabled us to carry out the research. To them, we give out heartfelt thanks and deepest appreciation.

Finally we would convey our thanks and admiration to all our classmates, friends and seniors, who had been supporting helping and encouraging us throughout the thesis project. We also wish to acknowledge MCS system administration for their help and support. We would like to express our gratitude and appreciation to faculty member Syed Ali Abbas Zaidi, for his help and support all along.

Table of Contents

List of Figures

Figure Number **Page**

List of Tables

Table Number **Page**

1 Introduction

1.1 Overview

Many facilities charge per vehicle entry for usage. Such facilities include parking areas, highways, bridges or tunnels. It is common to have staff at the checkpoints to control the payment of fees or to have machines if a set fee is required. The purpose of this research is to investigate a suitable way to read and recognize the registration plate from a photograph of a vehicle. An automated system could then be implemented to control the payment of fees. In fact, any situation requiring the automatic control of identification of a vehicle with a license plate number may be a potential application for such a system. At roads there is a need of a system which can detect stolen cars and also suspicious vehicles for security purposes.

1.2 Background

In recent days security has become very vital issue for an organization and for us. The heavy traffic on roads is also a serious problem for us the car lifting is also very popular these days. These problems can be minimized through the use of a License Plate Detection and Recognition System.

1.3 Application Domain

License Plate Recognition has a wide range of applications since the license number is the primary, most widely accepted, human readable, mandatory identifier of motor vehicles. License Plate Recognition provides automated access of the content of the license plate for computer systems managing databases and processing information of vehicle movements. Bellow we indicated some of the major applications, without the demand of completeness.

1.3.1 Parking

One of the main applications of LPR is parking automation and parking security: ticket less parking fee management, parking access automation, vehicle location guidance, car theft prevention, "lost ticket" fraud, fraud by changing tickets, simplified, partially or fully automated payment process, amongst many others.

1.3.2 Access Control

Access control in general is a mechanism for limiting access to areas and resources based on user's identities and their membership in various predefined groups. Access to limited zones, however, may also be managed based on the accessing vehicles alone or together with personal identity. License plate recognition brings automation of vehicle access control management, providing increased security, car pool management for logistics, security guide assistance, event logging, event management, keeping access diary, possibilities for analysis and data mining.

1.3.3 Motorway Road Tolling

Road Tolling means, that motorists pay directly for the usage of particular segment of road infrastructures. Tolls are a common way of funding the improvements of highways, motorways, roads and bridges: tolls are fees for services. Efficient road tolling increases the level of related road services by reducing travel time overhead, congestion and improve roadways quality. Also, efficient road tolling reduces fraud related to non-payment, makes charging effective, reduces required manpower to process events of exceptions. License plate recognition is mostly used as a very efficient enforcement tool, while there are road tolling systems based solely on license plate recognition too.

1.4 License plates

The system for catching speed violators depends primarily on recognizing the front side license plates. This section describes the characteristics for a standard license plate.

1.4.1 Dimensions

All license plates must be either rectangular or quadric shaped. The dimensions of the plate can vary, depending on the vehicle type, but the most common license plate is the rectangular shaped shown in Figure 1.1

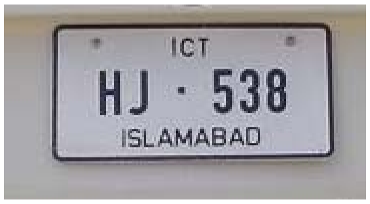

Figure 1-1: A standard license plate

1.4.2 Layout

The front side license plate on all vehicles is either white enclosed by a black frame. All standard license plates have two capital letters ranging from A to Z, followed by digits ranging from 0 to 9. All characters and digits are written in black on white license plates.

1.4.3 Mounting and material

The front side license plate must be mounted horizontally, and in upright position seen from the side of the car. In other words, the textual part must face directly forward. The plate may not be changed in shape, decorated or embellished, and the plate may not be covered by any means. Bolts used for mounting may not affect the readability of the plate and must be painted in the same color as the mounting point on the plate.

1.5 Delimitation

This section concludes the chapter by summing up the delimitations for the system. The delimitations are grouped by subject.

1.5.1 License plate formats

This project does not consider square or customized plates. The vast majorities of license plates currently in use are of the standard rectangular shape and are not customized.

1.5.2 Quality of decisions

As described in Section 1.2 a key element in making the proposed system useful is, that in cases of doubt when identifying the characters, which is when the probability of the best guess being correct is below some threshold, the system should refuse to make a decision. This issue is not addressed in this project and instead the best guess is always used as the final result.

Chapter 2

2 License Plate Extraction

2.1 Introduction

Before isolating the characters of the license plate in the image, it is advantageous to extract the license plate.

This chapter presents three different extraction strategies. First the theory behind each method is developed, then the strengths and weaknesses are summarized and finally, in the last section of this chapter, the three strategies are combined. The strategies are Hough transform, template matching and region growing. The common goals of all these strategies are, given an input image, to produce a number of candidate regions, which with high probability contains a license plate.

The purpose of this chapter is not to develop three competing methods, since it is unlikely that one method will prove better in all cases. Instead the goal is to have the strengths of each method complement the weaknesses of the other methods and in this way improve the overall efficiency of the system.

The methods presented in this chapter are all based on several assumptions concerning the shape and appearance of the license plate. The assumptions are listed in the following:

1. The license plate is a rectangular region of an easily discernable color
2. The width-height relationship of the license plate is known in advance
3. The orientation of the license plate is approximately aligned with the axes
4. Orthogonality is assumed, meaning that a straight line is also straight in the image and not optically distorted.

The first two assumptions are trivially fulfilled, since the license plate is bright white and the size is known. The last two assumptions are based on the fact that the camera should always be aligned with the road. This is necessary in order to reduce perspective distortion.

Several other objects commonly found in the streets of urban areas fit the above description as well. These are primarily signs of various kinds. This should be taken into account when finding locations for placing the camera in the first place. Even if some of these objects are mistakenly identified, this is not a great problem since later processing steps will cause these to be discarded.

2.2 Hough transform

This section presents a method for extracting license plates based on the Hough transform. As shown in Figure 2.1 the algorithm behind the method consists of five steps.

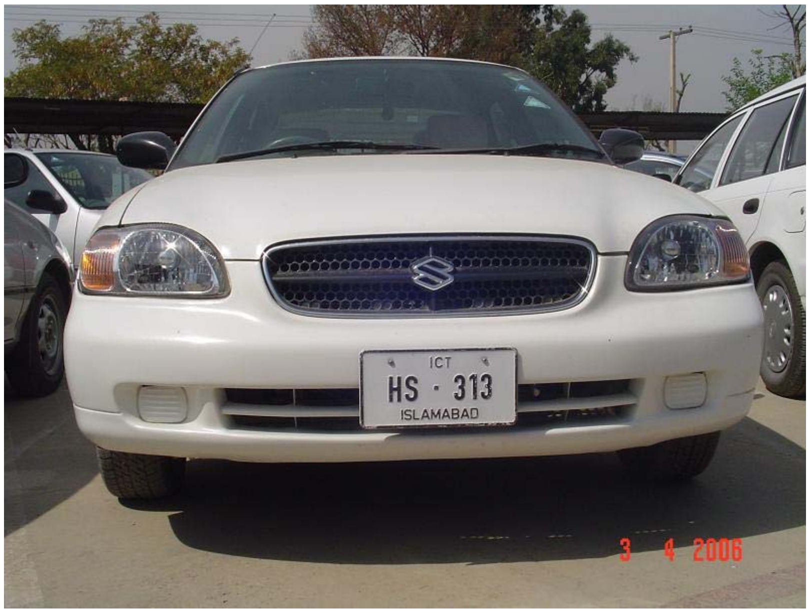

Figure 2-1: Source image

The first step is to threshold the gray scale source image. Then the resulting image is passed through two parallel sequences, in order to extract horizontal and vertical line segments respectively.

The first step in both of these sequences is to extract edges. The result is a binary image with edges highlighted. This image is then used as input to the Hough transform, which produces a list of lines in the form of accumulator cells. These cells are then analyzed and line segments are computed.

Finally the lists of horizontal and vertical line segments are combined and any rectangular regions matching the dimensions of a license plate are kept as candidate regions. This is also the output of the algorithm.

2.2.1 Edge detection

Before computing the edges, the source image is thresholded. The choice of optimal threshold value is highly dependent on lighting conditions present when the source images were taken. The value chosen in this project was based on the training images.

The first step is to detect edges in the source image. This operation in effect reduces the amount of information contained in the source image by removing everything but edges in either horizontal or vertical direction. This is highly desirable since it also reduces the number of points the Hough transform has to consider.

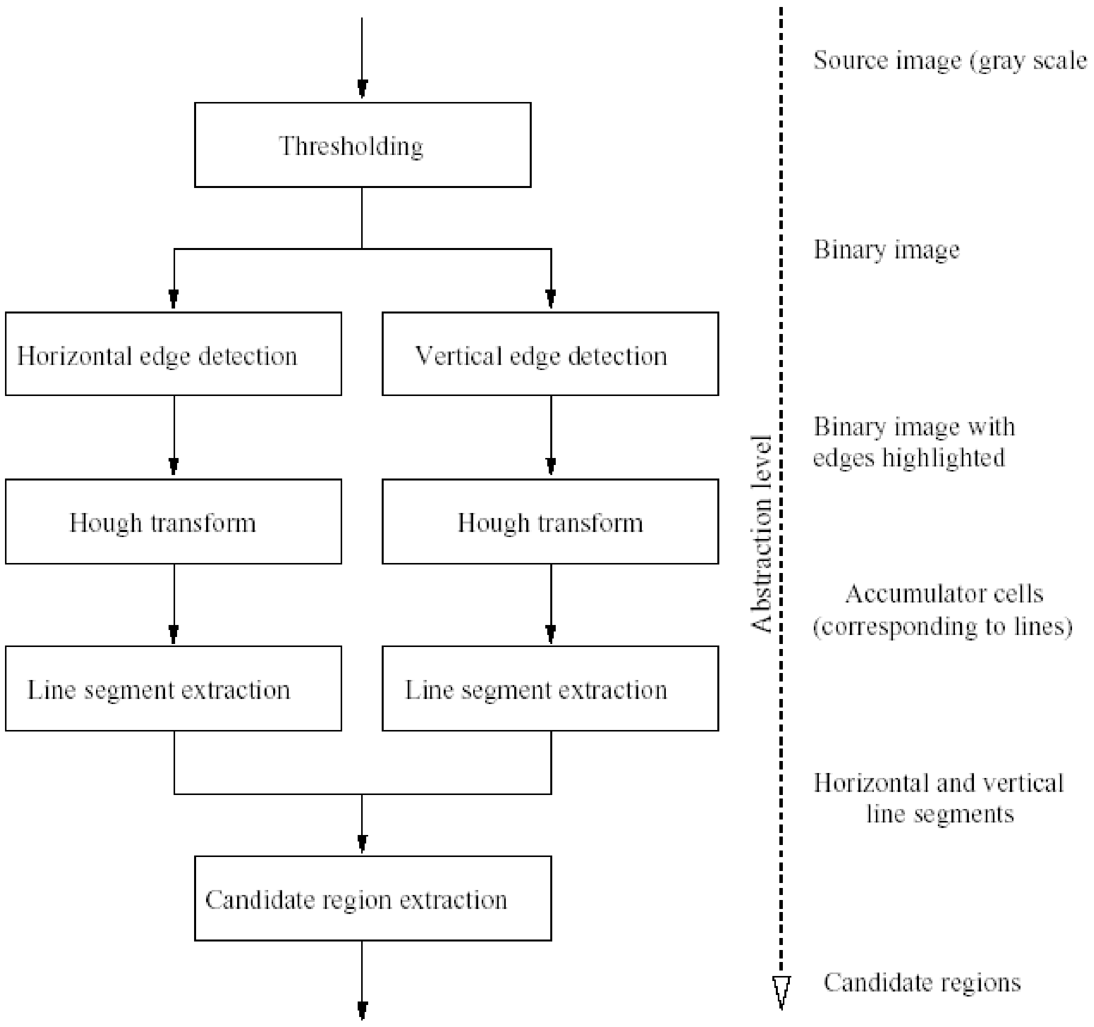

Figure 2-2: Overview of the Hough transform method

The edges are detected using spatial filtering. The convolution kernels used are shown in Equation (2.1). Since horizontal and vertical edges must be detected separately, two kernels are needed.

$$R_{horizontal} = \begin{bmatrix} -1 \\ +1 \end{bmatrix} \qquad R_{vertical} = \begin{bmatrix} -1 & +1 \end{bmatrix} \tag{2.1}$$

The choice of kernels was partly based on experiments and partly because they produce edges with the thickness of a single pixel, which is desirable input to the Hough transform.

Figure 2.3 shows the two kernels applied to the inverted thresholded source image. The image is inverted to make is easier to show in this report. Since the two filters are high pass filters, which approximates the partial derivatives in either horizontal or vertical direction, a value of positive one in the resulting image corresponds to a transition from black to white and a value of minus one corresponds to the opposite situation. Since it is of no importance which transition it is, the absolute value of each pixel is taken before proceeding.

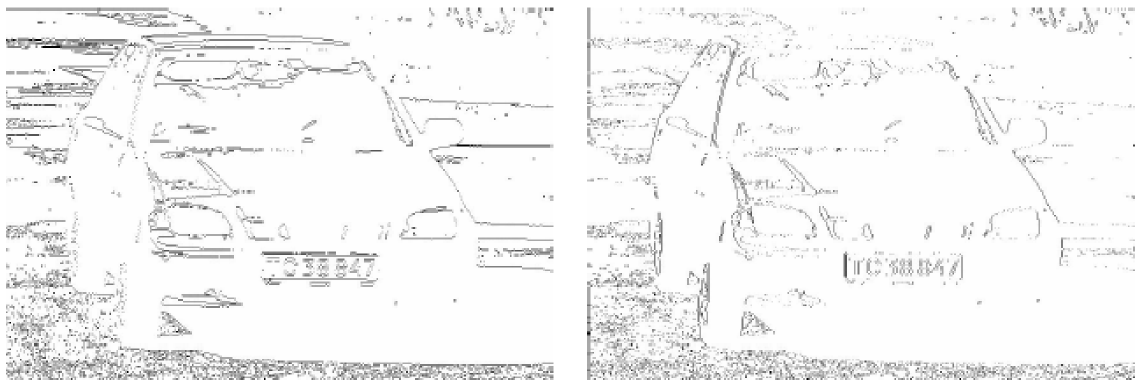

Figure 2-3: Horizontal and vertical edges

2.2.2 Line detection using Hough transform

The Hough transform is a method for detecting lines in binary images. The method was developed as an alternative to the brute force approach of finding lines, which was computationally expensive.

A brute force approach to finding all lines in an image with n points would be to form all lines between pairs of points and then check if the remaining points were located near the lines. Since there are roughly n(n-1)/2 of these lines and roughly n comparisons per line this is an $O(n^3)$ operation.

In contrast the Hough transform performs in linear time. The Hough transform works by rewriting the general equation for a line through (x_i, y_i) as:

$$y_i = ax_i + b \Leftrightarrow \qquad (2.2)$$

$$b = -x_i a + y_i \qquad (2.3)$$

For a fixed (x_i, y_i), Equation (2.3) yields a line in parameter space and a single point on this line corresponds to a line through (x_i, y_i) in the original image. Finding lines in an image now simply corresponds to finding intersections between lines in parameter space.

In practice, Equation (2.3) is never used since the parameter a approaches

∞ as the line becomes vertical. Instead the following form is used:

$$x\cos\theta + y\sin\theta = \rho \qquad (2.4)$$

In Equation (2.4) θ the parameter is the angle between the normal to the line and the x-axis and the ρ parameter is the perpendicular distance between the line and the origin. This is also illustrated in Figure 2.4. Also in contrast to the previous method, where points in the image corresponded to lines in parameter space, in the form shown in Equation (2.4) points correspond to sinusoidal curves in the $\rho\theta$ plane.

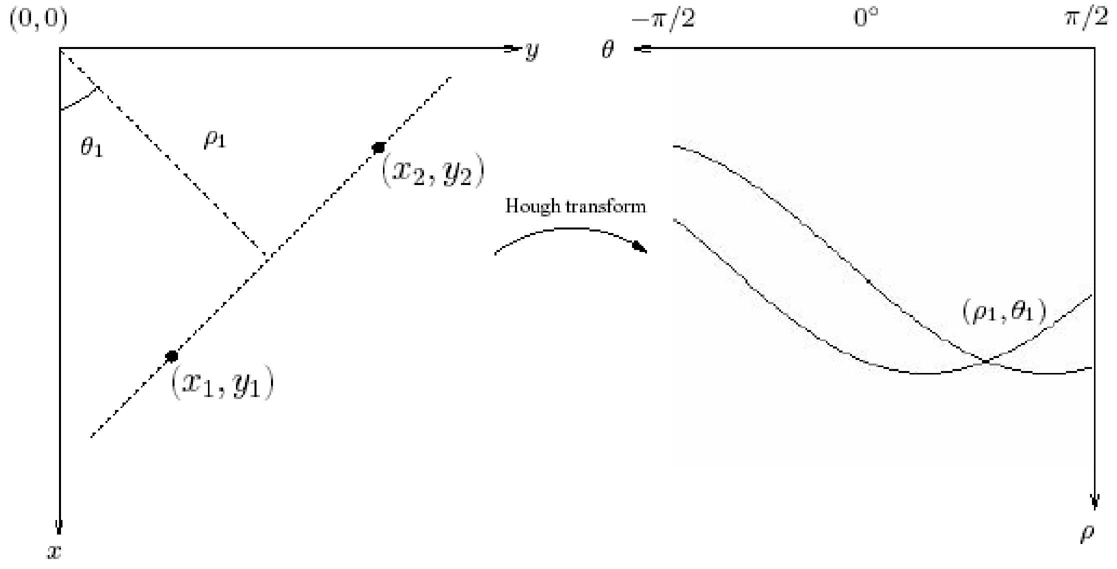

Figure 2-4: Example image and Hough transform

As shown in Figure 2.4, the range of θ is $\pi/2$ and for an image with a resolution of $w \times h$ the range of ρ is 0 to $\sqrt{w^2 + h^2}$.

Figure 2.4 also shows two points, (x_1, y_1) and (x_2, y_2), and their corresponding curves in parameter space. As expected, the parameter of their point of intersection in parameter space corresponds to the parameters of the dashed line between the two points in the original image.

The algorithm behind the Hough transform is now straight forward to derive. First the accumulator array is cleared to zero. Then for each point in the edge image iterate over all possible values of θ and compute ρ using Equation (2.4). Finally for each computed (ρ, θ) value, increment the corresponding accumulator cell by one. Since the algorithm iterates over all points it is clear that it performs in $O(n)$ time.

After the algorithm has iterated over all points, the accumulator cells contain the number of points which contributed to that particular line. Finding lines is then a matter of searching the accumulator array for local maxima.

2.2.3 Line segment extraction

The Hough transform in its original form as proposed in supports only line detection as opposed to line segment detection. A slight modification of the algorithm makes it possible to extract line segments.

The key is simply to remember which points in the original image contributed to which lines. Instead of only storing the number of contributing points in the accumulator cells, references to the points are also kept. The downside to this procedure is that in general for an image with n points and a resolution in θ of r_θ exactly $n \times r_\theta$ references must be stored. As mentioned above, the transform is only computed for a single value of θ, so r_θ is always 1 in this project, which makes this fact less of a problem.

Finding line segments is now a matter of iterating over each cell and searching for line segments. This is done by first sorting all the points according to their position along the line in question and then grouping points into line segments. Since it is known that the points are in order, grouping is most easily done by iterating over all points and for each points checking whether the next point in line is within some maximum distance. If it is, the line segment is extended to include the new point, otherwise a new line segment is started. In this project the maximum distance, or maximum gap length, was chosen as the value which gave the best results for the training images. Also, as some of the line segments are potentially very short, specifying a minimum line segment length is advantageous and helps reduce computation in later steps.

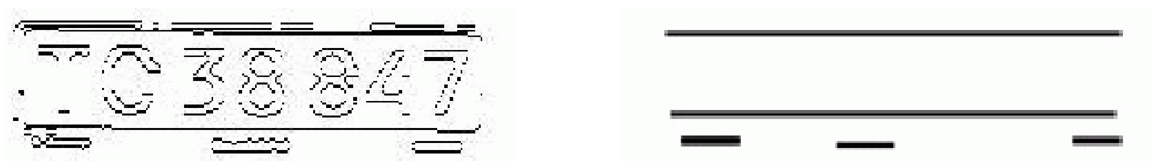

Figure 2-5: Example license plate and reconstructed line segments

Figure 2.5 shows an ideal case, where the longest line segments from the image have been extracted. The short line segments in the letters and in the top part have been eliminated.

The first step is sorting the points along the line. This is done by first finding the parameterized equation of the line. A cell with parameters (ρ_i, θ_i) corresponds to a line with Equation (2.5).

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos(\theta i + \pi / 2) \\ \sin(\theta i + \pi / 2) \end{bmatrix} t + \rho \begin{bmatrix} \rho i \cos \theta i \\ \rho i \sin \theta i \end{bmatrix} \tag{2.5}$$

Computing t for each point and sorting after t is now trivial. The result of this step is a list of line segments.

2.2.4 Candidate region extraction

The final step is to find the candidate regions, that is the regions believed to contain license plates. The input to this step is a list of horizontal line segments and a list of vertical line segments.

The procedure used is first to scan the lists for pairs of segments that meet the following requirements

1. They should start and end at approximately the same position.
2. The ratio between their average length and distance should equal that of a standard license plate.

The resulting two lists of regions, one with horizontal and one with vertical pairs are then compared and any region not contained in both lists are discarded. The remaining regions are the final candidate regions.

2.2.5 Strengths and weaknesses

This section discusses the advantages and disadvantages of the Hough transform method. The strengths are listed in Table 2.1 and the weaknesses in Table 2.2.

Strengths	**Explanation**
Scaling invariant	Since the algorithm does not look for regions of particular size, it is invariant to scaling of the license plate.
Relatively independent of license plate color	As long as the license plate is brighter than the surroundings, the plate is usually correctly extracted.

Table 2-1: Strength of the Hough transform method

Weaknesses	**Explanation**
Trouble detecting vertical lines	Vertical lines in the license plate are typically more than a factor four shorter than the horizontal lines and thus more susceptible to noise.
Finds more than just the license plate	All rectangular regions with dimensions equal to that of a license plate are identified, which is sometimes many. This makes it difficult to choose the correct candidate later.

Table 2-2: Weakness of the Hough transform method

2.3 Template matching

The main philosophy behind extraction through template matching is, that by comparing each portion of the investigated image to a template license plate, the actual license plate in the image is found as the region bearing the most resemblance to the template. A common way to perform this task is to use a cross correlation scheme.

2.3.1 Template construction

The way the template is constructed plays a significant role in the success of template matching. There are no rules for template construction but the template must be constructed in such a way, that it has all the characteristics of a license plate as it is presented in the source image.

Image preprocessing

When examining the source images, it is quite obvious that there are two major differences in the plates. The plates vary in size because of the fact that the cars are at different distances and the license plates vary in light intensity due to different lighting conditions when the images are taken. Both are issues that cannot be handled by the template construction.

A bit simplified, the goal is to make the plates look similar. This can be done by thresholding the image. The threshold value is not calculated dynamically because the image lighting is unpredictable and the different locations and colors of cars makes it impossible to predetermine the black (or white) pixel percentage needed for dynamic thresholding. Therefore the value giving the best result in a number of test images is chosen. Figure 2.6 shows two images, with very similar license plates, but different overall lighting, to demonstrate, that a dynamic threshold is hard to accomplish.

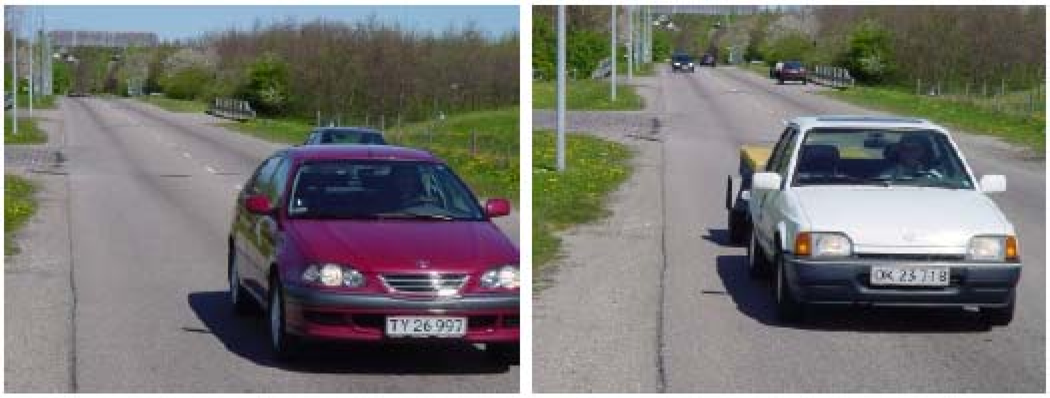

Figure 2-6: Images with different lighting

Also, a license plate has an edge, which on the processed image will appear to be solid black, and a bit thicker in the bottom. It is however not a good idea to add this feature to the template. Even a small rotation of the license plate means that the correlation would yield a big error.

2.3.2 Cross correlation

One of the more common ways to calculate the similarities between two images is to use cross correlation. Cross correlation is based on a squared Euclidean distance measure in the form:

$$d^2_{f,t}(u,v) = \sum_x \sum_y [f(x,y) - t(x-u, y-v)]^2 \tag{2.6}$$

Equation (2.6) is an expression for the sum of squared distances between the pixels in the template and the pixels in the image covered by the template. The value calculated represents that of pixel (u,v) in the correlated image as shown in Figure 2.7. The pixels near the edges are ignored as the correlation is only carried out when the entire template can be correlated. A correlation near the edges would mean that a pseudo-pixel value in the image for the area under the template exceeding the image would have to be assigned.

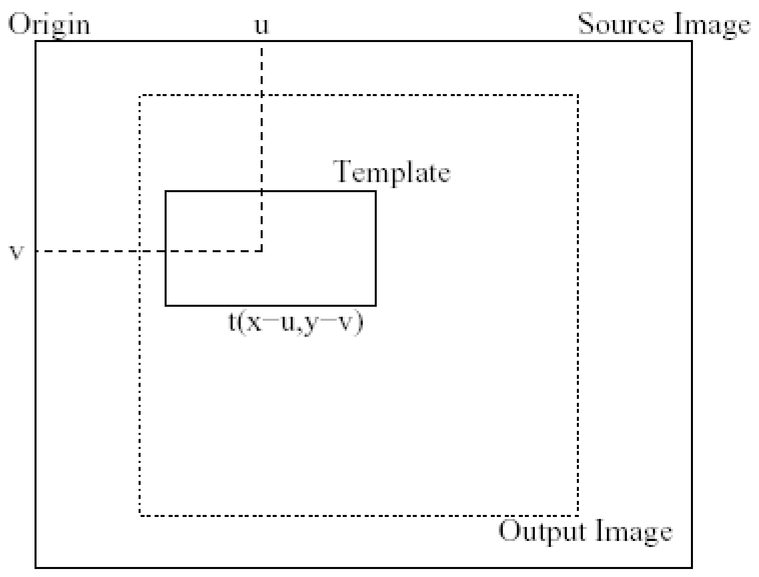

Figure 2-7: The layout of cross correlation

Expanding the expression for the Euclidean distance, d^2, produces:

$$d^2_{f,t}(u,v) = \sum_x \sum_y [f^2(x,y) - 2f(x,y).t(x-u,y-v) + t^2(x-u,y-v)] \qquad (2.7)$$

Examining the expansion, it is noted that the term $\sum_x \sum_y t^2(x-u,y-v)$ is constant, since it is the value of the sum of pixels in the entire template squared.

2.4 Region growing

This section will describe another method for finding uniform regions (such as a license plate) in an image. The basic idea behind region growing is to identify one or more criteria that are characteristic for the desired region. Once the criteria have been established, the image is searched for any pixels that fulfill the requirements. Whenever such a pixel is encountered, its neighbors are checked, and if any of the neighbors also match the criteria, both of the pixels are considered as belonging to the same region. Figure 2.8 visualizes these steps. The criteria can be static pixel values, or depend on the region that is being expanded.

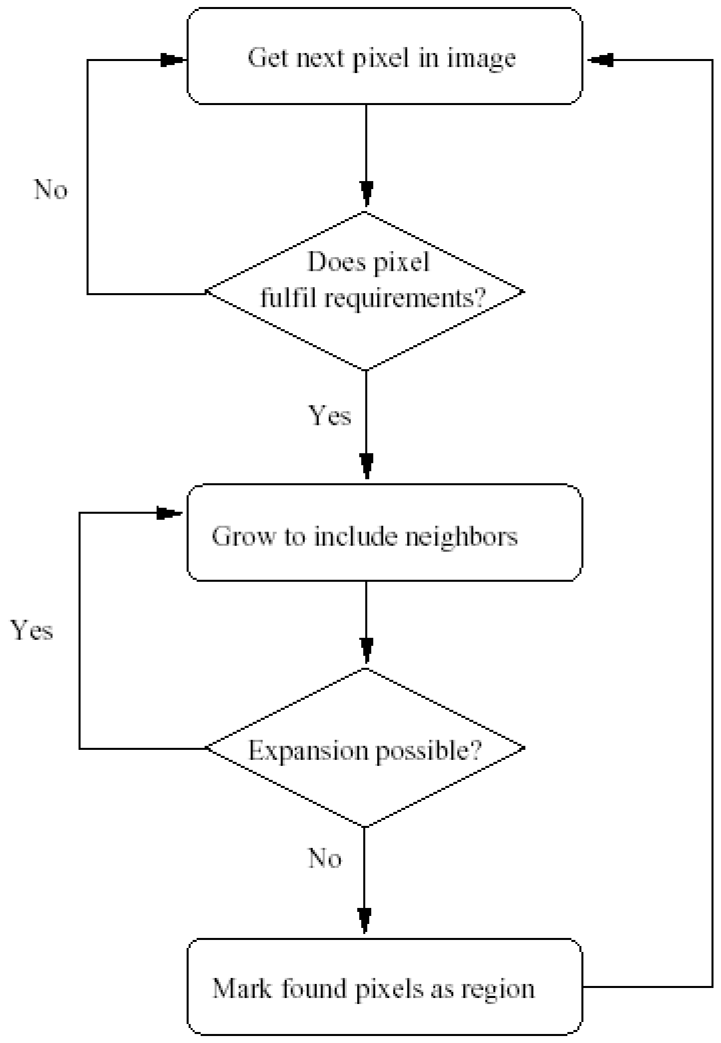
Get next pixel in image
No
Does pixel fulfil requirements?
Yes
Grow to include neighbors
Yes
Expansion possible?
No
Mark found pixels as region

Figure 2-8: The steps of region growing

2.4.1 License plate features

License plates are made from a reflecting material, they will usually appear brighter than the rest of the vehicle. Exceptions occur with white cars, but in general, looking at the brightness is an effective way of finding possible candidates for license plates. Of course, it has to be considered, that the characters inside the plate are black. This is helpful when a license plate has to be distinguished from e.g. other parts of a white vehicle. License plates also have well defined dimensions.

2.4.2 Preprocessing

Before searching through the image for any pixels bright enough to be part of a license plate, the image is prepared by converting it to binary. This is done, since color plays no role when looking for bright pixels. In performing this conversion, the threshold value is critical as to whether or not the license plate can be distinguished from its surroundings. On one hand, the threshold should be low enough to convert all pixels from the license plate background into white, on the other hand it should be chosen high enough to convert as much of the other parts of the image as possible into black. Figure 2.9 shows an example of such a binary image. The license plate appears as a bright rectangle, but there are several other white regions.

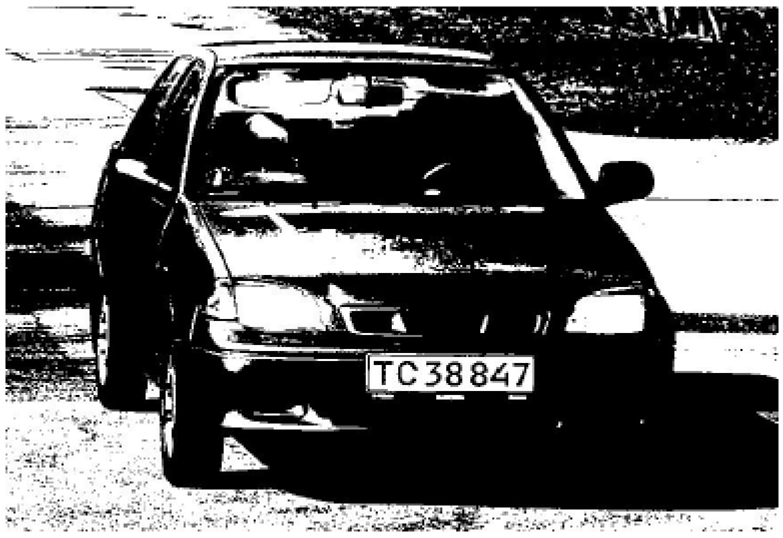

Figure 2-9: Binary image

A problem arises, since the overall brightness of the images is not known in advance, and therefore it is reasonable to select a relatively low threshold to guarantee that the background of the license plates are always converted into white. Then other criteria, such as the ratio mentioned above, will have to help in selecting the most probable candidate for a license plate, from among the list of white regions.

2.4.3 Extracting the regions

When using the region growing concept to extract regions with similar features, the most intuitive approach is to use recursion. As described in Section 2.4, each pixel is compared to the criteria (in this case only brightness), and whenever a pixel fulfills the requirements, all of its neighbors are also compared to the criteria. This method requires that all pixels that have been examined are marked to prevent the same region from being extracted multiple times.

This can be achieved merely by setting the value of all previously found pixels to black.

The nature of the region growing algorithm does not guarantee that a region satisfies the condition of being a square with certain dimensions. There are several solutions to this problem:

1. A region is transformed into the largest contained rectangle within its area.

This method is susceptible to noise, since a single black pixel along an otherwise white line of pixels, could prevent the rectangle from reaching the optimal size (see Figure 2.10).

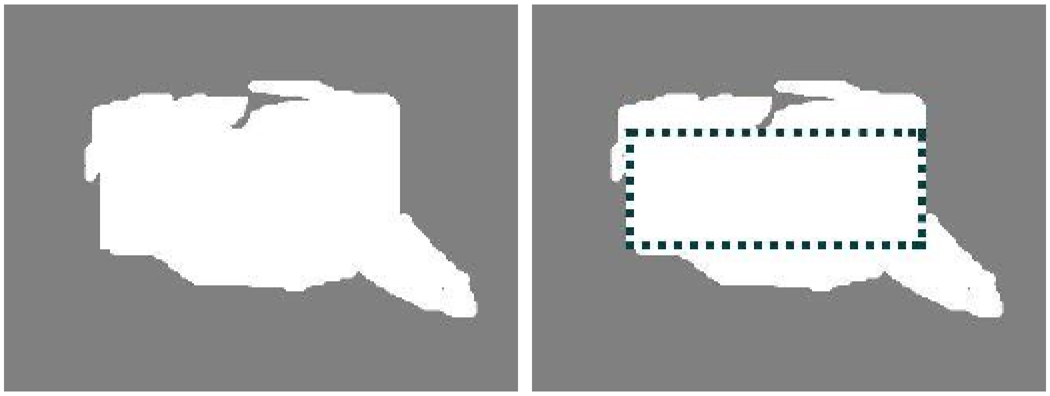

Figure 2-10: Largest contained rectangle

2. A region is transformed into a rectangle, based on maximum and minimum values of the pixels it contains.

This method reduces the effect of noise, but introduces the possibility of mistaking arbitrary white forms for a white square. It guarantees, however, that the entire license plate will be contained in a single region, even though the plate is not completely horizontal (see Figure 2.11).

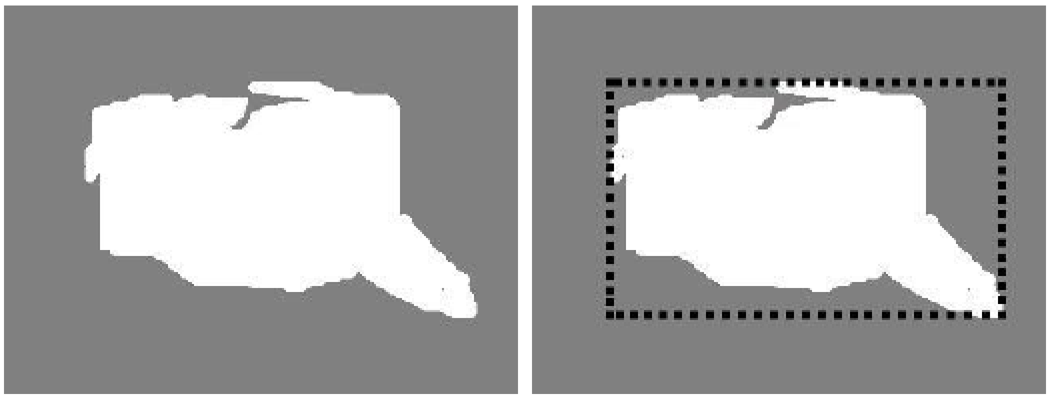

Figure 2-11: Rectangle for maximum and minimum value

Since the method of transforming a region into the largest contained rectangle is susceptible to noise, it is reasonable to assume that the second method will prove to give the best results.

2.5 Combining the method

In order to select which method to use and how to combine them, each method must be evaluated against the demands set by the system.

2.5.1 Scale invariance

The size of the license plate in the image varies and therefore the extraction method must be invariant to those changes. Hough transformation is only concerned with the horizontal and vertical lines in the image, and the ratio between them.

Region growing finds a bright region and again it is basically the ratio between width and height that is evaluated. Therefore both the Hough transformation and the region growing method are, as they should be, invariant to changes in license plate size.

Template matching is not scale invariant. Since template matching is based on a pixel by pixel comparison, any change in size makes the result unpredictable.

2.5.2 Rotation invariance

To reduce the number of needed calculations, the Hough transform was designed to find horizontal and vertical lines, based on the assumption that the license plate is aligned with the axes of the image. Because of this a license plate slightly rotated will not be extracted.

Region growing finds the bright areas regardless of orientation, and a small rotation does not change the height-width ratio noticeably. This means that the method is capable of extracting rotated plates.

Template matching is also seriously affected by rotation. If the plate is rotated even slightly, the outcome of the template matching is, as was the case with scaling, unpredictable.

2.5.3 Lighting invariance

The surroundings and weather conditions make it difficult to use any form of dynamic thresholding, when converting the image into binary. The success rate of both template matching and region growing suffers because of it. Hough transform is however much less sensitive to changes in the lighting conditions under which the picture was taken. The reason is, that while both template matching and region growing thresholds the image with a predetermined value, the Hough transformation is applied to an image, on which edge detection has been performed. As long as the lines in the image are detectable by the edge detection algorithm, the Hough transformation does not care how bright or dark the image is.

2.6 Summary

This chapter introduced three methods for finding candidate regions for the license plate. While region growing and Hough transform seem to be viable algorithms, the correlation scheme suffers from a scaling problem, which prevents it from being a real alternative. Also, it was demonstrated how the selection of the most probable candidate takes place, with three different methods that complement each others weaknesses. One of these was correlation, which proved to be a good discriminator, when sorting out regions that does not contain a license plate.

Chapter 3

3 Character Isolation

3.1 Introduction

To ease the process of identifying the characters, it is preferable to divide the extracted plate into seven images, each containing one isolated character. This chapter describes several methods for the task of isolating the characters.

Since no color information is relevant, the image is converted to binary colors before any further processing takes place. Figure 3.1 shows the ideal process of dividing the plate to seven images containing a character each.

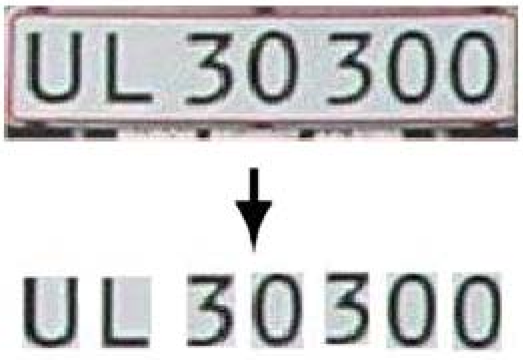

Figure 3-1: Isolating characters

3.2 Isolating the characters

There are several approaches for finding the character bounds, all with different advantages and disadvantages. The following section describes three different approaches and how these methods are combined to provide a very robust isolation.

3.2.1 Static bounds

The simplest approach is to use static bounds, assuming that all characters are placed approximately at the same place on every license plate as shown in Figure 3.2. The license plate is simply divided into seven parts, based upon statistical information on the average character positions.

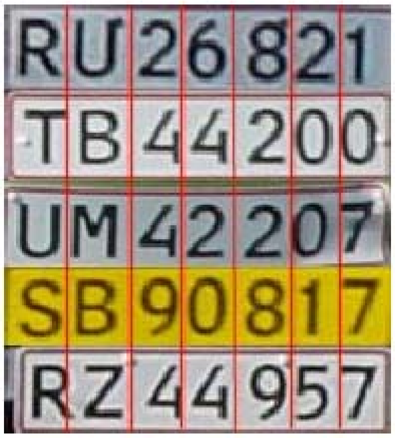

Figure 3-2: Example showing statically set character bounds

The advantage of this method is the simplicity, and that its success does not depend on the image quality (assuming that the license plate extraction is performed satisfactory). Its weakness is the fairly high risk of choosing wrong bounds and thereby making the identification of the characters difficult. This risk is directly proportional to the quality of the license plate extraction output.

The first character on the lowest plate in Figure 3.2 shows an example output from the plate isolation, where a portion of the mounting frame was included. Another weakness is that the method only separates the characters instead of finding the exact character bounds.

3.2.2 Pixel count

Instead of using static bounds, a horizontal projection of a thresholded plate often reveals the exact character positions, as demonstrated in Figure 3.3. The approach is to search for changes from valleys to peaks, simply by counting the number of black pixels per column in the projection. A change from a valley to a peak indicates the beginning of a character, and vice versa.

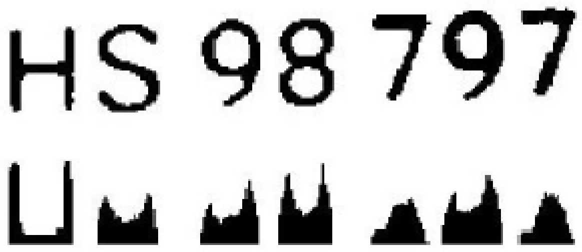

Figure 3-3: Horizontal projection of the actual character

Depending on the quality of the license plate extraction, and the success of removing the frame, this method is very useful, since it is independent of the character positions. The downside is that this method is very dependent upon image quality and the result of the license plate extraction.

3.2.3 Connected components

Another method is to search for connected components in the image. The easiest way to perform this task is on a binary image, using simple morphology functions.

The method for extracting connected components in a binary image is to search for a black pixel. When such a pixel is found, it assumed that it is a part of a component and therefore the basis for the an iterative process, based on Equation (3.1)

$$x_k = (x_{k-1} \oplus B) \cap A \qquad k = 1,2,3,... \qquad (3.1)$$

Where x_k represents the extracted component, A is the source image and B is a structuring element of size 3×3 indicating 8-connectivity neighboring.

x_0 is the first black pixel from where the iteration starts.

The algorithm starts by creating a new image x_0, only containing the first black pixel from where they it begins illustrated as the black structure in Figure 3.4. In the first step of the iterative algorithm, x_0 is dilated using the structuring element B. This means that the area of interest is expanded from the first black pixel, to all of its neighbors. The outcome x_1 of the first iteration is then the similarities from the dilation of x_0 and the original image A, found by an AND operation. In other words, all the neighbors of the first black pixel that are also black belong to the component searched for.

In the next iteration x_1 is dilated, and the result is the similarities between the dilation and the original image. This iterative process continues until the resulting component equals the previous, $x_k = x_{k-1}$ meaning that resulting component no longer differs from the previous. The first two iterations of the process is illustrated in Figure 3.4.

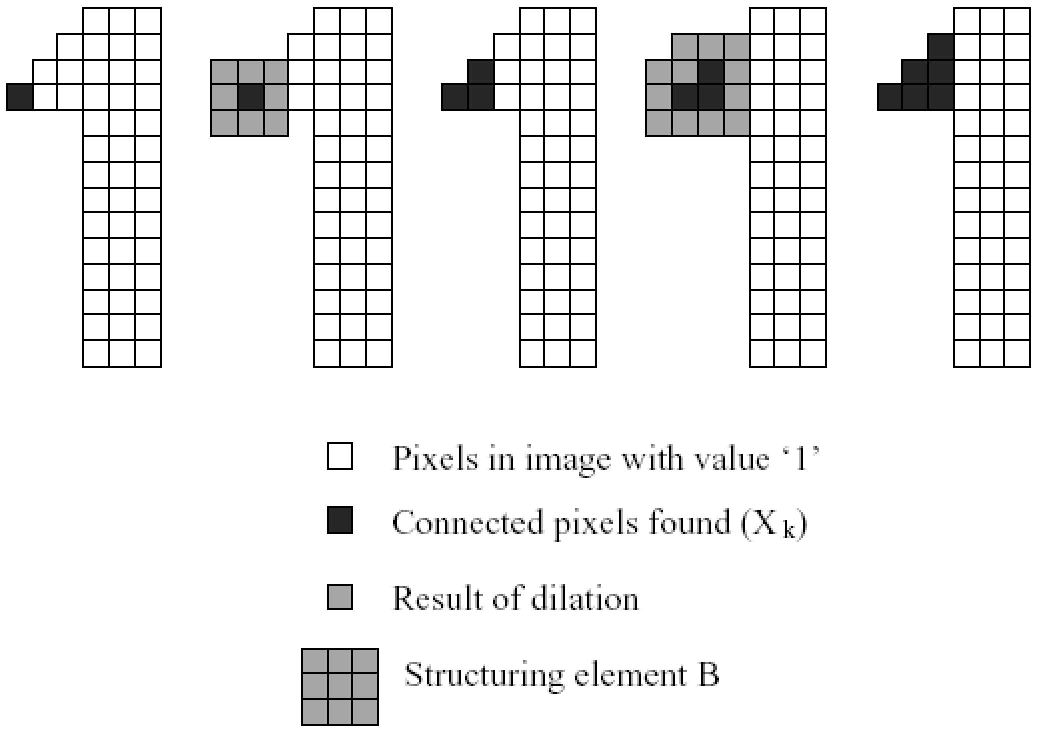
Pixels in image with value '1'
Connected pixels found (X_k)
Result of dilation
Structuring element B

Figure 3-4: An iteration in the process of finding connected component

When a connected component has been found, it is tested whether it fulfills the size specifications for a character, and thereby sorting out unwanted parts, for example bolts or dirt.

The advantages of this method is that it its independent of the image rotation and that the bounds found are very precise. Its disadvantages are that it requires a good image quality and a good conversion from the original image to the binary image, to avoid making two or more characters appear as one connected region. Small gaps in the characters can also cause the method to fail.

3.2.4 Improving image quality

Variations in image quality and license plate conditions can affect all of the methods mentioned previously. The following sections describe how the image quality can be improved.

Isolating the plate

The image received from the extraction often contains more than just the license plate, for example the mounting frame as in Figure 3.1. This frame can be removed by further isolation of the actual license plate.

The isolation of the license plate from any superfluous background is performed using a histogram that reflects the number of black pixels in each row and in each column. In most cases, projecting the amount of black pixels both vertically and horizontally reveals the actual position of the plate. An example of the projections is shown in Figure 3.5.

A simple but effective method for removing the frame is based on the assumption, that the vertical projection has exactly one wide peak created by the rows of the characters. Therefore the start of the widest peak on the vertical projection must be the top of the characters, and the end of the peak the bottom of the characters. It is expected that the horizontal projection has exactly seven wide peaks and eight valleys (one before each character, and one after the last character).

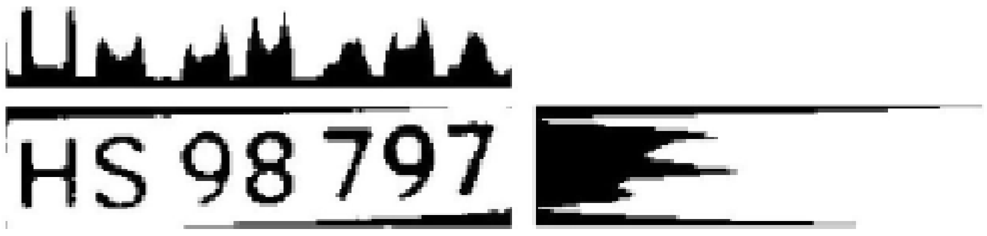

Figure 3-5: Horizontal and vertical projection

The success of this method depends on the assumption that the plate is horizontal. In some cases the method will result in a small part of the frame being left back in parts of the image, for example if the angle of the plate is too large, although depending on the thickness of the remaining frame, it is still possible to separate the characters.

Dynamic threshold

If the original image is dark or the license plate is dirty, the binary image created from the standard threshold value can be very dark and filled with unwanted noise. There are various methods to eliminate this problem, the threshold used when the original image is converted from color to binary, is calculated dynamically.

Removing small objects

A different approach is to remove the unwanted pixels from the binary image. Knowing that we are looking for seven fairly large regions of black pixels, it can be useful to process a number of erosions on the image before searching for the bounds, and thereby removing unwanted objects, for example bolts.

3.2.5 Combined strategy

	Static bounds	Pixel count	Connected components
Extra Edges	✗	✓	✓
Bad threshold	✓	✗	✗
Noise	✓	✓	✗

Table 3-1: Strengths of different methods

To achieve the best possible isolation, a combined strategy can be used to improve the probability of success. Table 3.1 shows how the three methods complement each other. While static bounds are immune to threshold values and image noise, it is very dependent upon whether all edges have been eliminated. On the other hand, when using pixel count the edges can easily be distinguished from characters, but the threshold value is crucial to its success. Finally, finding the connected components is susceptible to noise as well as a bad threshold. However, the connected components are less likely to be disturbed, since the weaknesses towards threshold and noise are rather small. The table should demonstrate that it is possible to combine the three methods, so that none of the three potential difficulties remain a problem.

Figure 3.6 shows how the different methods are combined, and the method is summarized below:

1) First convert the image into binary colors using a dynamic threshold to achieve the best result.
2) Try the connected-component method to see if seven good components can be found.
3) If the first attempt is unsuccessful, try to isolate the plate by cutting away edges and try the connected components method again.
4) If still unsuccessful, try the pixel-count method to search for the character bounds on the horizontal projection.
5) If this method also fails, use static bounds as the final option.

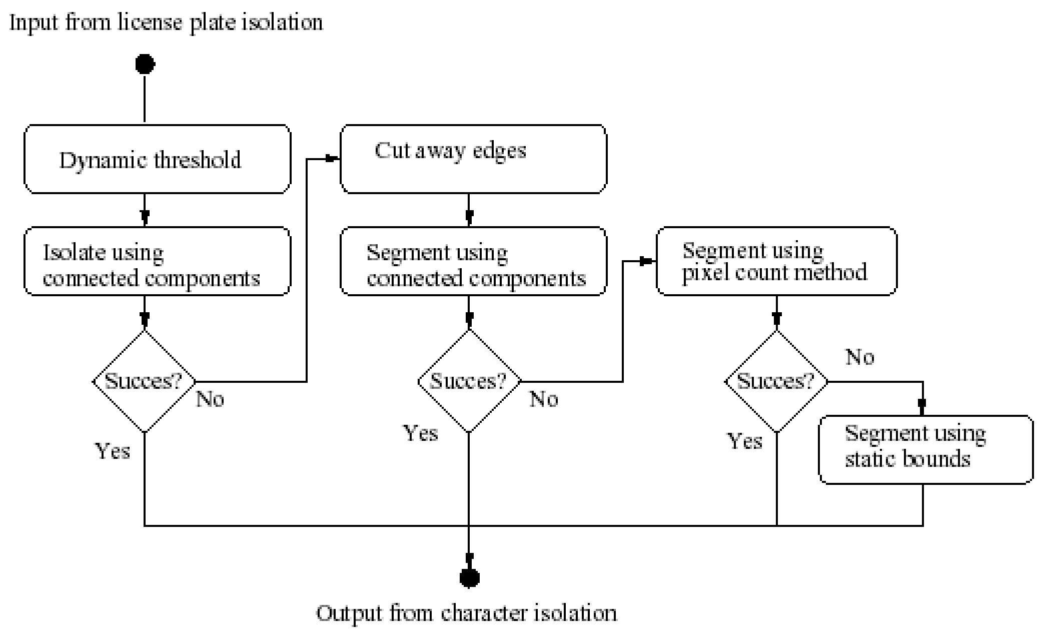
Input from license plate isolation
Dynamic threshold
Isolate using connected components
Succes?
No
Yes
Cut away edges
Segment using connected components
Succes?
No
Yes
Segment using pixel count method
Succes?
No
Yes
Segment using static bounds
Output from character isolation

Figure 3-6: Composition of character isolation procedure

A connected component is always performed as the first attempt, since it has a very good probability of success if the input image does not contain too much noise in the form of edges or similar black regions. If the connected components method fails, even after an attempt to cut away the edges, pixel count is performed as the first alternative. The static bounds are not utilized, until all other options have been exhausted. The reason is, that the static bounds are very reliant on the fact, that there are no vertical edges, and that the plate is totally aligned with the axes.

3.3 Summary

In this chapter several approaches to isolating the characters from an input image containing the entire license plate, was described. None of the methods were capable of providing reliable results on their own, due to the varying input image quality, but a combination of the methods ensures a very robust isolation scheme.

Chapter 4

4 Character Identification

4.1 Introduction

After splitting the extracted license plate into seven images, the character in each image can be identified. Identifying the character can be done in a number of ways. In this chapter, methods for this will be addressed.

First, a solution based on the previously discussed template matching (Section 2.3), will be presented. Thereafter, a method based on statistical pattern recognition will be introduced, and a number of features used by this method will be described. Then, an algorithm for choosing the best features called SEPCOR will be presented. After finding a proper set of features, means of selecting the most probable class is necessary. For this purpose, Bayes decision rule will be introduced, along with theory on discriminant functions and parameter estimation. Finally the methods will be compared against each other and it will be examined if there is anything to be gained from combining the two.

4.2 Template matching

This section presents how the isolated characters can be identified by calculating the normalized correlation coefficient. Previously it was investigated how template matching could help in the extraction of the license plates from the input images, and therefore the theory behind template matching can be reused. The conclusion then was that it was unable to perform the task by itself, but was useful in combination with other methods. Searching through a large image with a small template could mean running the algorithm hundreds of thousands of times, whereas if a single similarity measurement of two same-sized images is needed, the algorithm would have to run only once.

The idea behind an implementation of a correlation based identification scheme is simple. Two template pools, one consisting of all the possible values for the letters, and one of all the values of the digits, are constructed. Once the license plate has been cut into the seven characters, each image containing a single character is evaluated in turn in the following way. The normalized correlation coefficient between the image of the character and the appropriate template pool is computed. The template that yields the highest coefficient indicates what character is depicted in the input images.

4.3 Statistical pattern recognition

Having looked at identifying characters using template matching, the focus is now turned to statistical pattern recognition. Several methods for pattern recognition exist, but one of the more classical approaches is statistical pattern recognition, and therefore this approach has been chosen for identifying the characters. For simplicity, the focus will only be on the images containing digits, but the methods are completely similar for the letters.

When using statistical pattern recognition, the first task is to perform feature extraction, where the purpose is to find certain features that distinguish the different digits from each other. Some of the features that are found might be correlated, and an algorithm called SEPCOR can therefore be used to minimize the number of redundant features. When an appropriate number of uncorrelated features have been found, a way of deciding which class an observation belongs to is needed. For this purpose Bayes decision rule will be introduced, and finally an expression for the classification will be proposed.

4.4 Features

All of the features are extracted from a binary image because most of the features require this. The area and the circumference of the digits are the simplest features to distinguish. These features are not sufficient, because there are different digits with approximately the same area and circumference, e.g. the digits `6' and `9'. To distinguish in such cases, the number of endpoints in the upper half and lower half of the image are taken into account.

Here it is assumed that the digit `6' has one endpoint in the upper half and zero endpoints in the lower half, and the endpoints of the digit `9' are the reverse of a `6'. To distinguish between the digits `0' and `8', it is not possible to use the endpoint feature because they both have zero endpoints. Here the number of connected compounds is an appropriate feature. The number of compounds in the digit `8' is three, whereas a `0' has only two compounds.

Furthermore the value of each pixel is chosen as a feature. A final feature is the area of each row and column, meaning the number of black pixels in each of the horizontal or vertical lines in the image.

The features are listed below:

1) Area
2) Area for each row
3) Area for each column
4) Circumference
5) End points in upper/lower half
6) Compounds
7) The value of each pixel

Most of the features require that the characters have the same size in pixels, and therefore the images with the characters are initially normalized to the same height. The height is chosen because there is no difference between the height of the characters, whereas the widths of the different digits differ, e.g. a `1' and an `8' have the same height but not the same width. The following describes the methods for extracting each feature.

4.4.1 Area

When calculating the area of a character, assuming background pixels are white and the character pixels are black, the number of black pixels are counted. In area for each row the numbers of black pixels in every horizontal line are counted. As seen in Figure 4.1, these vertical projections are distinct for many of the digits. The horizontal projections are more similar in structure, with either a single wide peak, or a peak in the beginning and end of the digit. Although more similar they will still be used to distinguish between the digits.

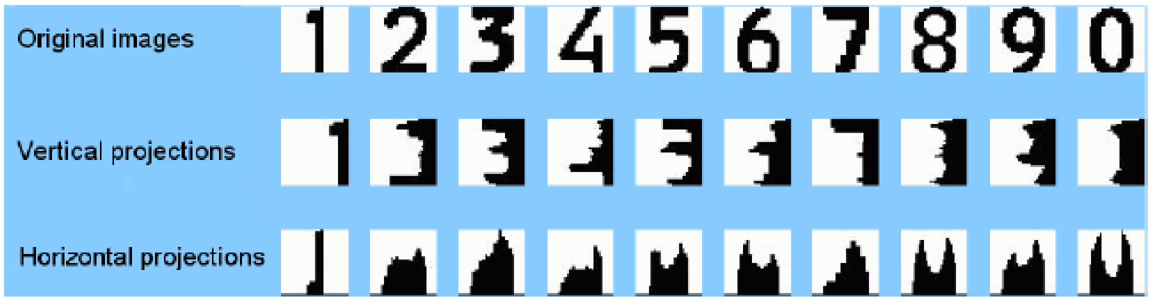

Figure 4-1: Projection of all the digits

4.4.2 End points

The number of end points is also a feature that distinguishes some of the digits from each other. It is also clear that the locations of these end points vary, and this can help in distinguishing between the digits. A `6', for instance, has only a single end point in the upper half, and none in the lower. Although the positions also vary horizontally, there would be problems when deciding whether the end points of a `1' are in the left or the right half of the image. Therefore, only the vertical position of the end points is considered in the implementation of the system.

The number of end points is found from a skeleton of the character. To obtain the skeleton a thinning algorithm is used and the result is a one pixel wide skeleton. The end points of the character are the pixels that have only one connected 8-neighbor.

The thinning algorithm iteratively deletes the edges of the character, until a one pixel wide skeleton is reached. The algorithm must fulfill that it does not delete end points and that it does not break connections. Breaking connections will result in additional end points, and therefore this is quite an important demand to the algorithm. In order to get the skeleton of the character, the algorithm is divided into two parts. The first part deletes edges in east, south or the northwest corner. The second part deletes edges in west, north or the southeast corner.

During the processing of the two parts, pixels that satisfy the conditions listed below are flagged for deletion. The deletion is not applied until the entire image has been processed, so that it does not affect the analysis of the remaining pixels.

The first part of the algorithm deletes pixels if all of the following conditions are satisfied.

(a) $2 \leq N(p1) \leq 6$

(b) $Z(p1) = 1$

(c) $p2.p4.p6 = 0$

(d) $p4.p6.p8 = 0$

$N(p1)$ is the number of neighbors (depicted in Figure 4.2) of $p1$ with the value one, this condition ensures that end points $(N(p1) = 1)$ and non-border pixels $(N(p1) > 6)$ are not deleted. $Z(p1)$ is the number of 0-1 transitions in the ordered sequence of $p2, p3, p4 .... p8, p9, p2$. An example of this can be seen in Figure 4.3, where $Z(p1) = 2$, it is a one pixel wide connection and in this situation $p1$ should not be deleted because this would break the connection, and because of condition (b) pixel $p1$ is not deleted. Conditions (c) and (d) satisfy that the pixel is placed in east, south or the northwest corner, see the gray pixels in Figure 4.4b.

p9	p2	p3
p8	p1	p4
p7	p6	p5

Figure 4-2: The neighbors of $p1$

The second part of the algorithm deletes pixels if condition (a) and (b) combined with (c) and (d) are satisfied.

$$p2.p4.p6 = 0$$
$$p4.p6.p8 = 0$$

0	1	0
0	p1	0
0	1	0

Figure 4-3: Number of 0-1 transition is 2, $Z(p1) = 2$

Condition (c) and (d) satisfy that the pixel is placed in west, north or the southeast corner; see the gray pixels in Figure 4.4c.

An example of using the thinning algorithm is depicted in Figure 4.4. The gray pixels in are those marked for deletion. a) shows the original images, b) is the result of applying part 1 one time and c) is the output of using part 2 once, d) is part 1 once more, and e) is the final result of the thinning algorithm.

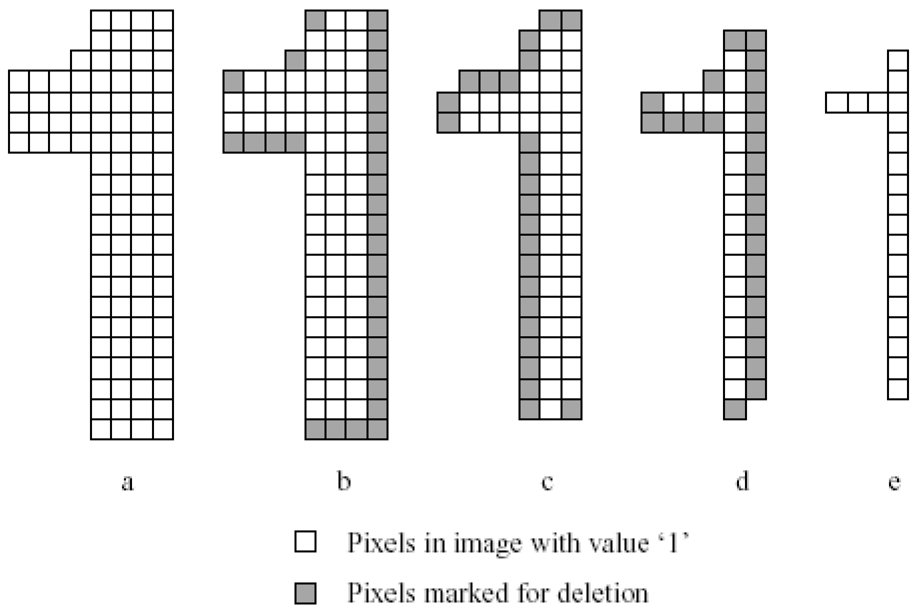

Figure 4-4: The steps of thinning algorithm

One iteration of the algorithm consists of:

1. Applying part one, and flag pixels for deletion.
2. Delete lagged pixels.
3. Applying part two, and flag pixels for deletion.
4. Delete flagged pixels.

The algorithm terminates when no further pixels are flagged for deletion.

When using the thinning algorithm even small structures are present in the final skeleton of the image, an example of this can be seen in Figure 4.5.

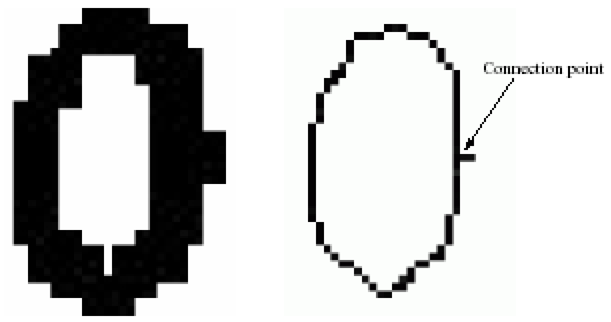

Figure 4-5: Result of thinning algorithm

Normally a `0' does not have any end points, but in this case it gets one end point. The way of finding end points is therefore slightly modified. Structures that are less than 3 pixels long before reaching a connection point are not taken into account. Then the digit `0' in Figure 4.5 has no end points, which was expected.

4.4.3 Circumference

Another possible feature to distinguish between characters is their circumference. As it can be seen in Figure 4.6, a `2' has a somewhat larger circumference than a ` 1’.

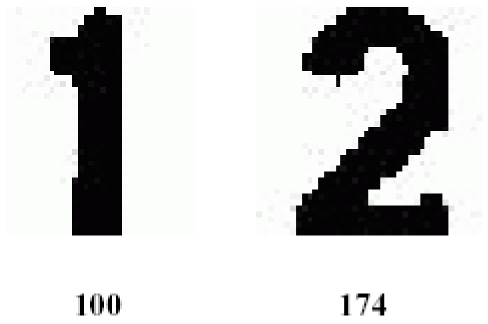

Figure 4-6: Two digits with different circumference

To find the circumference of a character, the pixels on the outer edge are counted. This is done by traversing the outer edge of the character, and counting the number of pixels, until the start position has been reached. The circumference is very dependent upon the selected threshold value, but relatively resistant to noise.

4.4.4 Compounds

Each character consists of one or more connected compounds. A compound is defined as an area that contains pixels with similar values. Since the image is binary, a compound consists of either ones or zeros. The background is only considered a compound if it appears as a lake within the digit. Figure 4.7 shows that a `0' has two compounds, while a `1' has only one.

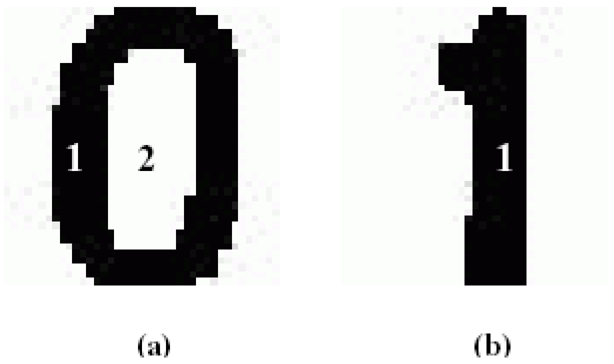

Figure 4-7: (a) shows two compounds in '0', and (b) the one in '1'

To find the number of compounds, a Laplacian filter is applied to the character. The Laplacian operator is defined as:

$$L[f(x,y)] = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{4.1}$$

This leads to the following digital mask:

$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Convolving the original binary image of the character with this mask, the result is an image with black background, and the edges represented by a one pixel thick white line, see Figure 4.8.

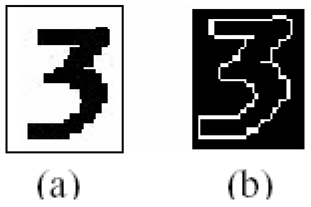

Figure 4-8: Applying Laplacian filter to a '3' (a) is the original and (b) is the resulting image

The number of compounds is then found by counting the number of connected white lines. Since the Laplacian filter is very sensitive to noise, white lines with lengths below a certain threshold are discarded. As alternative method for finding connected compounds, the method used when isolating the characters could also be used. The alternative method was described in Section 3.2.3.

4.4.5 The value of each pixel

The value of each pixel is used as a feature. In this way, the feature based identification resembles correlation based identification. When the Euclidean distance between a sample and a class is calculated, the difference is, that in the template based identification the correlation measure is normalized, which it is not in the feature based identification. Additionally, in the feature based identification more features are being used, such as the number of end points. In both cases, the distance is measured from a mean calculated from the training set.

4.5 Decision theory

After having acquired a sample in the form of a feature vector with values for all the previously described features, the sample must be classified as belonging to one the ten different classes. To do this, a statistical classifier is used. In this chapter, Bayes classifier will be presented and evaluated as a discriminant function used to divide the feature space.

The use of the Mahalanobis distance measure as an alternative to the Euclidean distance is based on the assumption, that the data set is normally distributed. Why this is a necessary assumption will be discussed, and whether or not the data set used in this project is normally distributed will be investigated by making various tests.

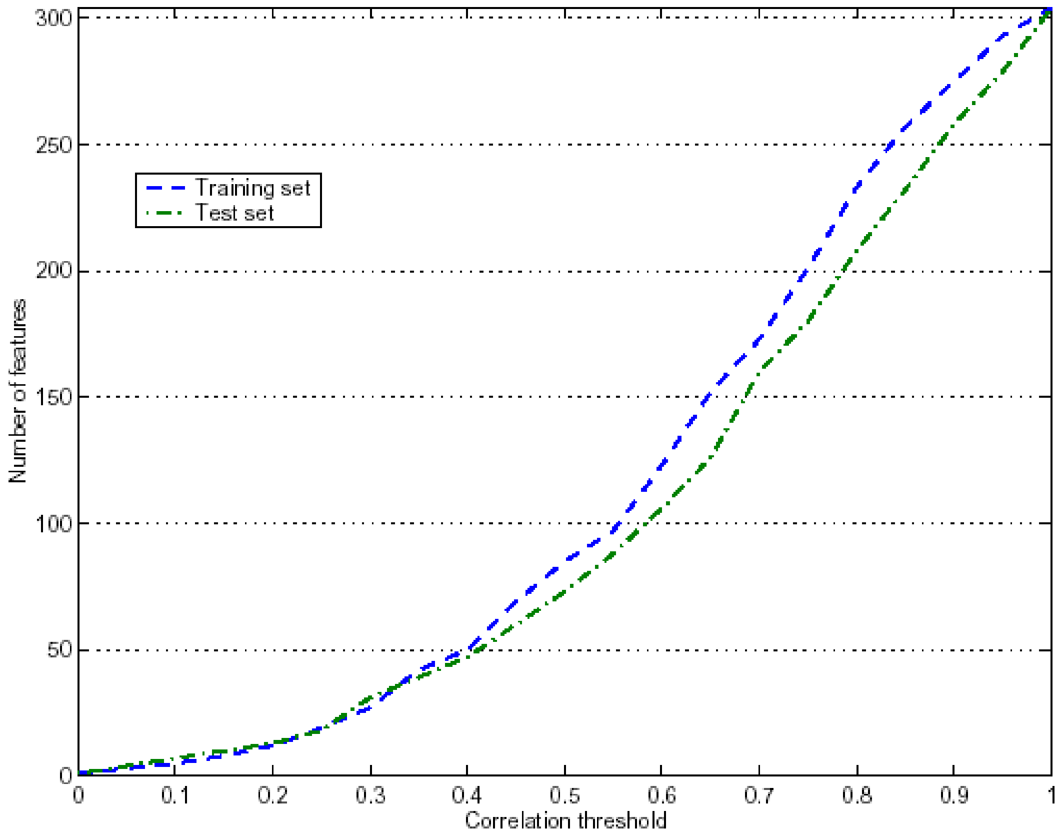

Figure 4-9: Number of features selected as a function of the maximum co-relation coefficient

When using the Bayes classifier it is necessary to know the class conditional probability density functions. Since the distribution of the data is assumed normal, this means that the parameters of this particular distribution has to be estimated. A method for doing this is also introduced.

4.5.1 Bayes decision rule

First, Bayes classifier is described:

$$P(\omega_j \mid \bar{x}) = \frac{p(\bar{x} \mid \omega_j).P(\omega_j)}{p(\bar{x})} \qquad (4.2)$$

in which

$$p(\bar{x}) = \sum_{j=1}^{s} p(\bar{x} \mid \omega_j).P(\omega_j) \qquad (4.3)$$

$\bar{x}$ is the feature vector of the sample being investigated. A finite set of classes, δ is defined as $\Omega = \{w1, w2, ...w_\delta)$. In this case, the classes represent the different characters that can occur on a license plate. Similar, a set of actions is declared as $A = \{\alpha 1, \alpha 2, ...\alpha_\delta)$. These actions can be described as the event that occurs when a character is determined to be e.g. of class $w2$. This implies assigning the correct character value to the given sample. The loss by performing the wrong action, meaning performing αi when the correct class is wj and is denoted by $\lambda(\alpha i \mid wj)$. The total loss by taking the wrong action can then be described as:

$$R(\alpha_i \mid \bar{x}) = \sum_{j=1}^{s} \lambda(\alpha_i \mid \omega_j).P(\omega_j \mid \bar{x}) \qquad (4.4)$$

Here is the probability that wj is the true class given $\bar{x}$. This can be used when having to make decisions. One is always interested in choosing the action that minimizes the loss. $R(\alpha_i \mid \bar{x})$ is called the conditional risk. What is wanted is a decision rule, $\alpha(\bar{x})$, that determines the action to be taken on a given data sample. This means, that for a given $\bar{x}$, $\alpha(\bar{x})$ evaluates to one of the actions in A. In doing so, the action that leads to the smallest risk should be chosen, in accordance with Equation 4.4.

Minimum-error-rate classification

As already stated, it is desirable to choose the action that reduces the risk the most. It can also be said, that if action αi corresponds to the correct class ω_j, then the decision is correct for $i = j$. The symmetrical loss-function is built on this principle:

$$\lambda(\alpha_i \mid \omega_X) = \begin{Bmatrix} 0 & i = j & \\ & & i, j = 1, ..., s \\ 1 & i \neq j & \end{Bmatrix} \qquad (4.5)$$

If the decision made is correct, there is no loss. If it is not, a unit loss is achieved, thus all errors cost the same. The average probability of error is the same as the risk of the loss function just mentioned, because the conditional risk evaluates to:

$$R(\alpha_i \mid \bar{x}) = 1 - P(\omega_i \mid \bar{x}) \qquad (4.6)$$

Where $P(\omega_i \mid \bar{x})$ is the conditional probability that α_i is the correct action.

So, in order to minimize the probability of error, the class i that maximizes $P(\omega_i \mid \bar{x})$ should be selected, meaning that ω_i should be selected if $P(\omega_i \mid \bar{x}) > P(\omega_j \mid \bar{x})$, when $i \neq j$.This is the minimum-error-rate classification.

4.5.2 Discriminant functions

Classifiers can be used to divide the feature space into decision regions. One way of partitioning is the use of discriminant functions, where a discriminant function $g_i(\bar{x})$ for each class is defined. Then the classifier assigns vector $\bar{x}$ to class ω_i if: $g_i(\bar{x}) > g_j(\bar{x})$ for all $j \neq i$.An example of dividing the feature space into decision boundaries can be seen in Figure 4.10, which accomplishes the minimum error-rate.

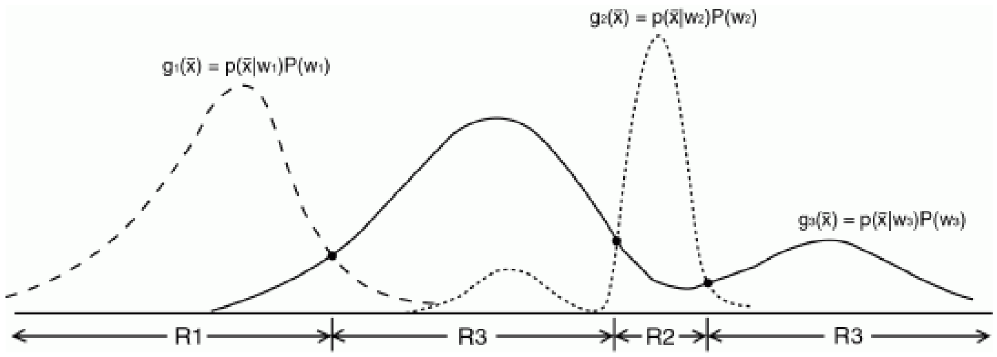

Figure 4-10: Decision regions

The feature vector $\bar{x}$ is assigned to the class corresponding to the region in which the vector is placed.

A Bayes classifier can also be represented in this way, where $g_i(\bar{x}) = P(\omega_i \mid \bar{x})$.The maximum discriminant function will then correspond to the a posteriori probability, see Equation 4.4. When looking at the a posteriori probability the discriminant function can also be expressed as $g_i(\bar{x}) = P(\omega_i \mid \bar{x})P(\omega_i)$, because the denominator is a normalizing factor and therefore not important in this context. Bayes classifier is therefore simply determined by $P(\omega_i \mid \bar{x})P(\omega_i)$.

The most commonly used density function is the multivariate normal density function, because of its analytical qualities. The univariate normal density function is given by:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 \right] \qquad (4.7)$$

Which is specified by two parameters; the mean μ and the variance μ^2. Normal distributed samples tend to cluster about the mean within 2μ.In the multivariate normal density function, the mean is instead represented as a vector, and the variance is represented as a matrix. The multivariate normal density function is given by:

$$p(\bar{x}) = \frac{1}{(2\pi)^{d/2} \mid \bar{\sum} \mid^{1/2}} \exp\left[-\frac{1}{2}(\bar{x}-\bar{\mu})^t \bar{\bar{\sum}}^{-1}(\bar{x}-\bar{\mu}) \right] \qquad (4.8)$$

Where $\bar{x}$ is a column vector with d entries, $\bar{\mu}$ is the mean vector also with d entries, and $\bar{\bar{\sum}}$ is a $d \times d$ matrix called the covariance matrix. Equation 4.8 can be abbreviated as $p(\bar{x}) \square N(\bar{\mu}, \bar{\bar{\sum}})$ where the mean and the covariance is calculated by:

$$\bar{\mu} = E[\bar{x}] \qquad (4.9)$$

$$\bar{\bar{\sum}} = E[(\bar{x}-\bar{\mu})(\bar{x}-\bar{\mu})^t] \qquad (4.10)$$

The covariance is calculated for each component in the vectors; x_i corresponds to the i'th feature of $\bar{x}$ and μ_i is the i'th component of $\bar{\mu}$ and σ_{ij} is the i-j'th component of $\bar{\bar{\sum}}$.

The entries in the mean and in the covariance matrix are calculated from:

$$\mu_i = E[x_i] \tag{4.11}$$

$$\sigma_{ij} = E[(x_i - \mu_i)(x_j - \mu_j)] \tag{4.12}$$

And when having n samples in the observation.

$$\mu_i = \sum_{k=1}^{n} \frac{x_{ik}}{n} \tag{4.13}$$

$$\sigma_{ij} = \sum_{k=1}^{n} \frac{(x_{ik} - \mu_i)(x_{jk} - \mu_j)}{n-1} \tag{4.14}$$

In the covariance matrix the diagonal, σ_{ii} and is the variance of feature i and the of-diagonal, σ_{ij} is the covariance of feature i and j. Samples drawn from a normal distributed population tend to gather in a single cloud. The center of such a cloud is determined by the mean vector, and the shape is dependent of the covariance matrix.

4.5.3 Test of normal distribution

In order to use the discriminant function method presented in the previous section, it is necessary to test whether or not the observations are normal distributed, because the method requires data to be so. Three methods have been used to determine if data is normal distributed. First, plotting the observation in a histogram and then visually determine if it is approximately normal distributed. Second, Matlab can be used to display a normal probability plot of the data, again with visual inspection used to determine the similarity with the normal distribution. The third method is a goodness of fit test, where a value is calculated to determine if data is normal distributed. In this section the circumference of the number `0' is used as an example for the three methods. Ideally all features should be investigated, and those that are not approximately normal distributed should be ignored.

The histogram plot is depicted in Figure 4.11, and it is seen that the graph approximately resembles a normal distribution.

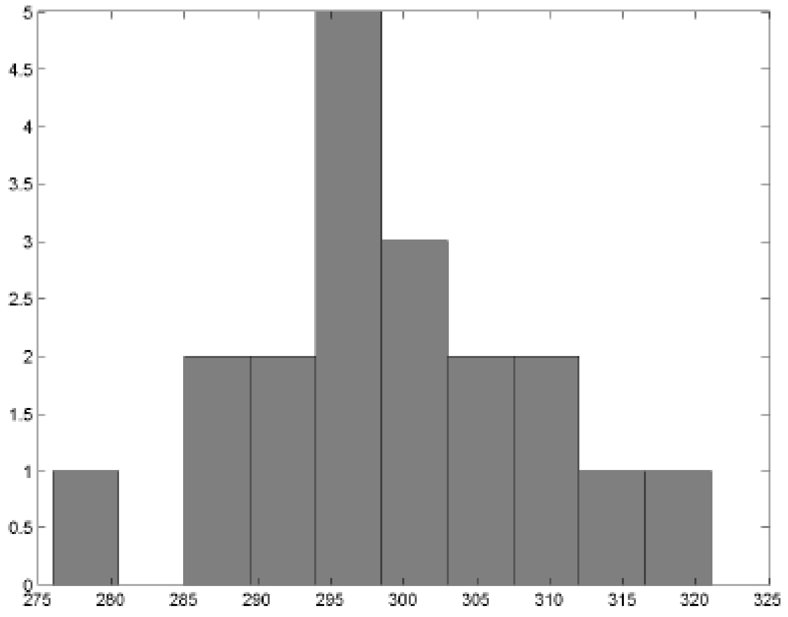

Figure 4-11: Histogram showing normal distribution

The Matlab plot displays true normal distributions as linear curves, and so it is expected, that the data from the circumference is approximately linear.

4.5.4 Parameter estimation

In the general case of using Bayes classification it is necessary to know both the a priori probabilities as well as the class conditional probability density functions. The problem is not the a priori probabilities, since these are usually easy to obtain, but instead the class conditional probability density functions. In practice, exact specifications of these functions are never directly available. Instead, what is available is a number of design samples which should be used for training the system plus evidence that these samples has some known parametric distribution.

It is assumed that the a priori probabilities of all the classes are equal. This reduces the problem of finding the class conditional probability density functions to a matter of estimating the parameters of a multivariate normal distribution that is the mean, $\bar{\mu}$ and the square covariance matrix $\bar{\bar{\sum}}$.

One method for parameter estimation is investigated. This method is called maximum likelihood estimation and takes a number of pre-classified sets of samples. $x_0, x_1, ...x_9$ as input. These samples are images of each of the numbers from the ten different classes.

4.6 Comparing the identification strategies

In order to decide which method to use it must be clear in which situations each method is preferable. This is done by identifying their individual strengths and weaknesses.

The strengths and weaknesses of template matching were listed in Table 2.3 and 2.4, and although directed towards license plate extraction, most of them do apply in more general terms.

Template matching is a strong similarity measure, when the size and rotation of the object to be compared is known. The algorithm is simple and fast, when it only has to run once on small images such as the isolated characters.

Feature based identification makes certain demands to the system in which it is to be used. The use of the involved classifiers demands that the a priori probabilities are accessible and it has to be possible to estimate the class conditional probability density functions.

Once these demands are met, the strengths and weaknesses are totally dependent on the features selected. If good features are available, the classification can be performed with only a few features, and depending on the speed of extracting these features, the method can be very fast.

Feature based identification is in many cases very sensitive to input quality.

Noisy images may hinder the correct extraction of certain features, which can be very disruptive for systems that base the identification on a small amount of features. The

amount of features in this system ensures that although one or two features fail, in the sense that they are wrongly extracted, the overall identification is still possible.
Template matching is also sensitive to noise and of course the accuracy decreases along with input quality, but not nearly as much as what might happen with feature identification.

4.7 Summary

In this chapter the two methods, template matching and statistical pattern recognition, for identifying characters were described. In connection with the statistical recognition. Mahalanobis distance was stated as a measure of identifying the characters, derived from Bayes decision rule. As an alternative measure, the Euclidean distance was used. A mean for testing, whether data is normal distributed was presented, and parameter estimation for the Bayes classifier is described. Finally, the two identification strategies were compared.

Chapter 5

5 Testing

5.1 Introduction

This chapter describes the test of the preprocessing step of isolating the characters in the license plate. The purpose of the chapter is to verify that the implementation of the isolation method described in Chapter 3 and Chapter 4 performs efficiently.

5.2 Character Isolation Test

The purpose of the method is to divide the extracted license plate into subimages, each containing one of the seven characters. A successful isolation fulfills all of the following criteria:

The plate must be divided into subimages.

None of the characters may be diminished in any way.

The sequence of the subimages has to be in the correct order. This means that the first character of the plate is the first of the subimages.

An example of an unsuccessful and a successful isolation is seen on Figure 5.1.

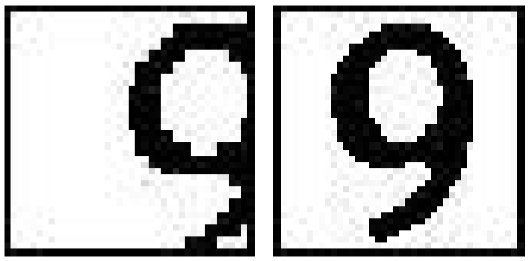

Figure 5-1: Example of unsuccessful and successful isolation

5.2.1 Test description

A series of input images is given to the algorithm and the success of the test simply depends on the resulting output images. The test has been performed using the connected component method, the pixel count method and the method using static bounds.

5.3 Character Identification Test

The final step in recognizing a license plate is identifying the single characters extracted in the previous steps. Two methods for doing this was presented earlier. The first that has been tested is the method based on statistical pattern recognition, and second the normalized cross correlation coefficient.

5.3.1 OCR Recognition

Some of the bitmaps for numbers 1 through 5 are shown in Figure 5.2. The test cases used for numbers 1 through 5 are also shown in Figure 5.3. The test cases used for numbers 1 through 5 are also shown. For each number, 64 inputs are generated for our neural model. Every bit that is turned on in a character's bitmap pattern has a value of 1 and each bit that is off has a value of 0. An input array of 64 1's and 0's will make up each character used in the training (and test) set. Consequently, the network design for this model must consist of 64 nodes in the input layer. The output layer has been designed with 10 output nodes—one node for each of the characters we wish to recognize. When a number is recognized from a set of inputs, the network will output a 1 at the appropriate output node. For this model, when the number 1 is recognized the first output node responds with a 1 and all other output nodes respond with a 0. The second output node responds with 1 when the number 2 is recognized and so on.

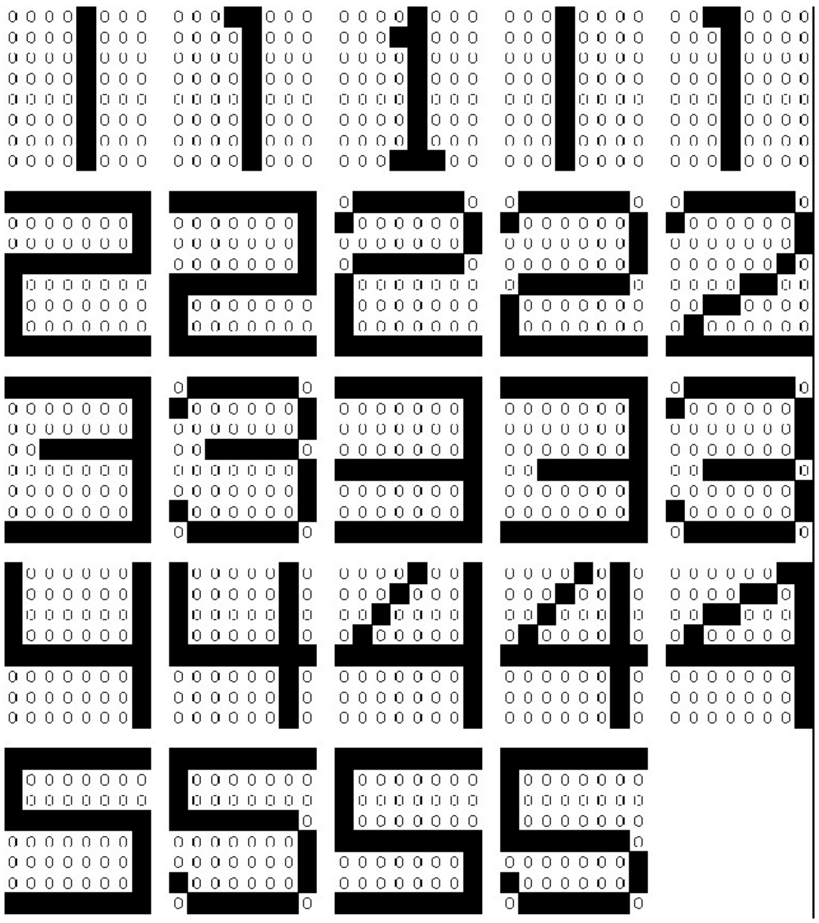

Figure 5-2: Sample of OCR training case.

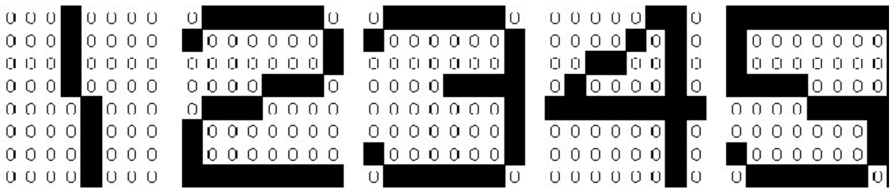

Figure 5-3: Sample of OCR test case.

5.4 Summary

To build a character recognition model for a full-featured OCR application, the model must be significantly more sophisticated than our sample shown here. There would likely be 100 or more output nodes to properly classify most of the common characters. Advanced capabilities like font type detection can be added

Chapter 6

6 Conclusion and Future Work

6.1 Introduction

The purpose of this project has been to investigate the possibility of making a system for automatic recognition of license plates, to be used by the police force to catch speed violators. In the current system, manual labor is needed to register the license plate of the vehicle violating the speed limit. The majority of the involved tasks are trivial, and the extensive and expensive police training is not used in any way. Therefore an automatic traffic control system, eliminating or at least reducing these tasks has been proposed.

We wanted to investigate the possibility of making such an automatic system, namely the part dealing with automatically recognizing the license plates. Given an input image, it should be able to first extract the license plate, and then isolate the characters contained in the plate, and finally identify the characters in the license plate. For each task, a set of methods were developed and tested.

For the extraction part, the Hough transform, and in particular the region growing method proved capable of extracting the plate from an image. Both of them locate a large number of candidate regions, and to select between them, template matching was utilized. Template matching, combined with height width ratio and peak-valley methods, provided a successful method of selecting the correct region.

The methods developed for isolating the characters proved very reliable.

The method for finding character bounds using an algorithm that searches for connected components proved to be the most useful, and combined with pixel count and static bounds, the method proved to be extremely successful.

For the actual identification process, two related methods were developed, one based on statistical pattern recognition and the other on template matching. Both methods proved to be highly successful, with feature based identification slightly better than template matching.

In order for the system to be useful, it should be able to combine the three different tasks, and to recognize the license plates in a high percentage, so the use of manual labor is reduced as much as possible. This implies, that the success rate for the individual parts should be close to 100 %. The results obtained are summarized in Table 6.1.

Mission	Success rate	Successful
Plate extraction	98.1 %	✓
Character isolation	100 %	✓
Character identification	98.9 %	✓
Overall performance	76.4 %	✓

Table 6-1: Main results Obtained

As can be seen, the individual parts perform very satisfactory, all with a success rate close to 100 %. The plate extraction succeeds in 98.1 % of the test images, and this is a very high success rate. The extraction fails in only two of the images. This is acceptable, since the extraction works in more than 98 % of the images, thereby fulfilling the criteria of this task.

The part of isolating the characters contained in the license plate succeeds in 100 % of the cases, and thus is very successful, achieving the goal set for this task.

Out of the isolated digits, 98.9 % were correctly identified. This is also a very high success rate, and it must be taken into account, that the wrongly identified digits originates from only two plates.

The overall performance is not as high as for the individual tasks, but still a large amount of license plates is correctly identified, namely 76.4 %.

In general, the conclusion of this report is that a system for automatic license plate recognition can be constructed. We have successfully designed and implemented the key element of the system, the actual recognition of the license plate.

6.2 Future Work

The algorithm can be easily modified to adjust to brightness and contrast in the environment and also the different characteristics of number plates at other places. Standards other than those in Islamabad can be incorporated in this algorithm.

Borderless or contrast less number plates can also be analyzed. Instead of searching for edges, complete body of a vehicle can be considered to find a number plate. Like the two red lights on both sides of the vehicle can determine location of the number plate by using mean of their positions.

Using a more professional database other than MS Access, databases like MS SQL, My SQL and Oracle if used will result in better results in different dimensions like CPU constraints, concurrent connections, memory constraints and security.

Appendix A

Contents of CD

This purpose of this appendix is to give an overview of the attached CD. Below a quick overview of the contents is displayed:

Directory: CD

- Referenced web pages and e books.
- Images
- Source code
- Documentation

All of the instructions required to install and use the program are included on the CD.

C.1 The program

The program was written in a Windows environment, since there are several well-documented image processing libraries available for Windows. Also, it was chosen to ease the process of creating a graphical user interface for testing purposes.
The code was written in Microsoft Visual studio .net, and the source code folder on the CD also contains the project files used when building the executable.

REFRENCES

[1] Rafael C. Gonzalez, Richard E. Woods, *Digital Image Processing*, Singapore; Addison-Wesley, 1993

[2] Moritz Stóring and Thomas B. Moeslund, *An Introduction to Template Matching*, Technical Report CVMT, 2001

[3] Richard O. Duda, Peter E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, 1973

[4] David C. Lay, *Linear Algebra and Its Applications*, Second Edition 2000, Addison-Wesley

[5] H Bunke, P S Wang, *Handbook of Character Recognition*,

[6] "Optical Character Recognition / Pattern Recognition"

http://www.qnetv2k.com/Qnet2000Manual/html/qnet65ym.htm

[7] "Multi-template GAT Correlation for Character Recognition"

doi.ieeecomputersociety.org/10.1109/ICDAR.2005.166

[8] "Classification and Learning for Character Recognition"

www.dsi.unifi.it/NNLDAR/Papers/01-NNLDAR05-Liu.pdf