A license plate recognition system : the design of a license plate recognition system for Dutch license plates

De la Haye, R.J.

Award date:
1998

A license plate recognition system
The design of a
license plate recognition system
for Dutch license plates.

Author: R.J. De la Haye

Student at:	Eindhoven University of Technology
ID-number:	331187
Coaches:	J.A. Hegt and N.A. Khan
Graduation professor:	W.M.G. van Bokhoven
Place:	Eindhoven University of Technology, Faculty of Electrical Engineering, Department S.E.S.
Date:	12-02-98

Summary

This report describes the design and implementation of a license plate recognition system for Dutch license plates. The system consists of five stages. The front-end is formed by a high-speed shutter camera and a frame grabber that delivers the digitised images of cars passing by. In the license plate segmentation step, the approximate position of the four corner points of a plate is indicated by hand for the time being. This stage has already been automated but is not available to us. The corner points may not correspond to a rectangle area due to the perspective view distortion. A bilinear transformation that makes use of bilinear grey value interpolation is applied to correct for this. The result is a rectangular license plate with a size of 180×40 pixels. Histogram stretching is applied to enhance the image for the character segmentation stage, which approximately segments the characters, based on the properties of the vertical projection of the license plate. The resulting characters are normalised with respect to contrast, intensity and size. Then the characters are projected onto a low-dimensional space with the help of the Hotelling transform. This transformation contains the relevant information that is needed to distinguish the characters. The transformation depends on a good segmentation, which is not guaranteed by the segmentation stage. Clues about the segmentation accuracy can be obtained by comparing the inverse Hotelling transformed result with the original character. If they differ significantly then the segmentation is probably bad. This leads to a much improved segmentation and thus a transformation that holds the needed information for the classification. The Hotelling transformed characters can be classified with different methods. A probabilistic neural network should result in the best performance, but it does not because of the limited amount of sample data. Classifying the transformed characters with the help of the Euclidean distance proved to result in the lowest misclassification and rejection rate. 545 plates were used to test the system. A misclassification rate of 0.4% was achieved with a rejection rate of 13%. Further development of the system, for which a number of recommendations are given, is expected to increase the system performance.

Contents

1. Introduction

The origin of this project lies at the Dutch ministry for traffic regulations. They announced to be interested in using license plate recognition systems in future. Philips responded to this by examining the possibilities of a license plate recognition system. Philips Industrial Vision, part of CFT[1], specialised in optical systems, image manipulation and character recognition is focussing the attention on such a system. This was also done earlier in the Kever [VII] project, which was only partially a success. It was too complex and expensive. Unfortunately not many details about this project are available to us. This time Philips Industrial Vision asked the faculty of Electrical Engineering at the Eindhoven University of Technology to do research on license plate recognition systems. At the university it was decided that this would make an interesting graduation project. I applied for this project because of my interest in artificial intelligence, text recognition and image processing.

Within the frame of the project "character recognition" of the faculty of Electrical Engineering at the Eindhoven University of Technology, neural and non-neural techniques are studied to recognise license plates. The problem of license plate recognition differs at many points from the problem of scanned text or hand-written text recognition. Here it is tried to find and implement the optimum approach that fits best to the problem of license plate recognition.

1.1 A system overview

The parts of a license plate recognition system that actually belong to the assignment can be pointed out in the system overview presented in figure 1.

[1] Centrum for Fabrication Technology

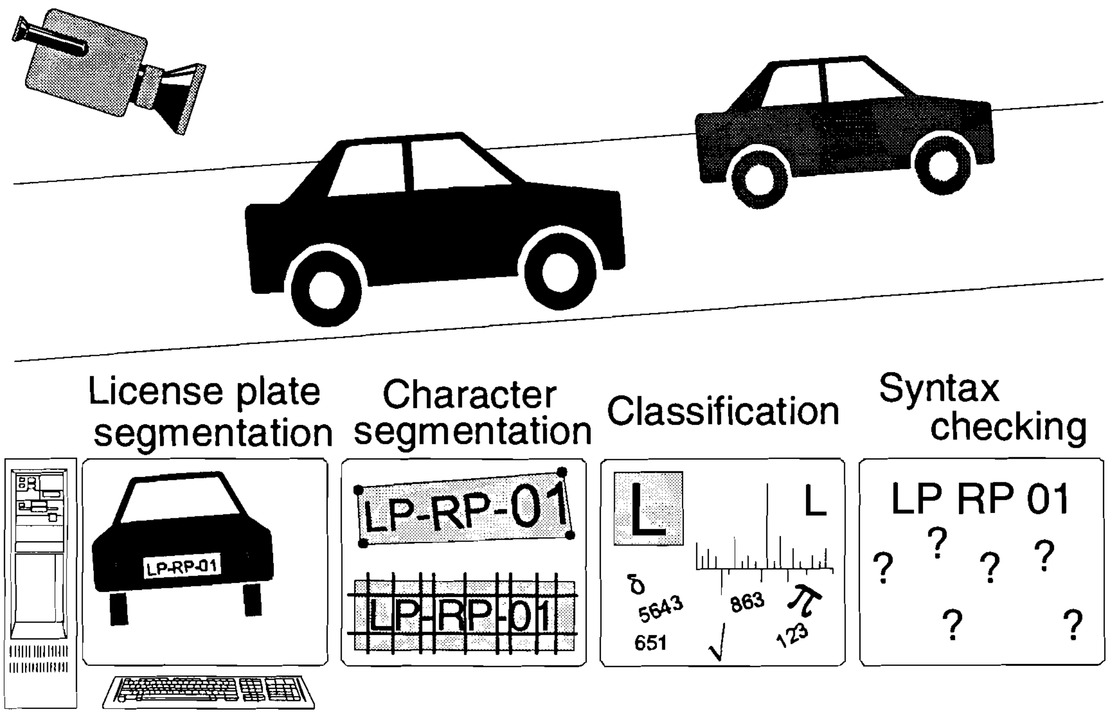


Figure 1. A system overview.

The system can be divided into five stages. All will roughly be discussed here.

A high-speed shutter camera and a frame grabber form the front-end of the system. The camera delivers images of cars passing by. A result like figure 2 is obtained after digitising an image with a frame grabber.

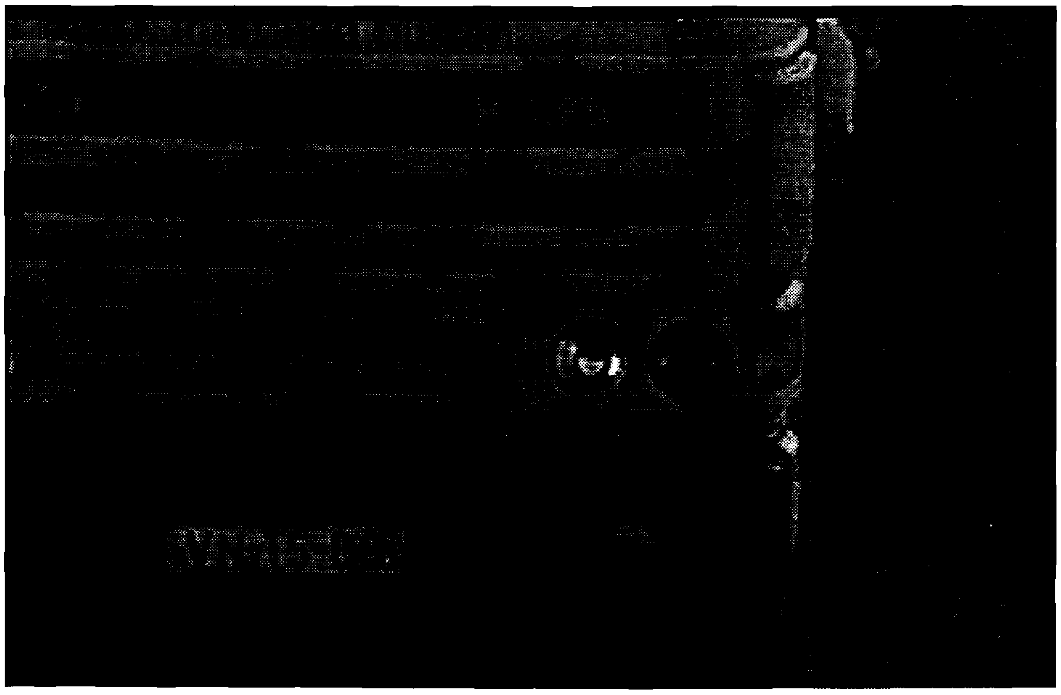


Figure 2. A sample image of the front of a truck.

This image is taken of the front of a truck, but it is preferred to take them of the back of a vehicle. The optical resolution of the images is assumed to be 439 × 510 pixels. The location of the license plate can be calculated from the digitised images. This is done in the license plate segmentation stage. The license plate segmentation stage of the system that was implemented by CFT was supposed to be used for this, but it is unavailable to us at the time, so no experiments can be done with it. Only some specifications are available. It is assumed that the system can find the corner points of the license plate in an image with a precision of about two pixels. These two pixels refer to the dimensions of the input image of 439 × 510 pixels. An image with four corner points indicated approximately is the starting point of the assignment.

The area described by the four corner points is transformed to a rectangular area of fixed dimensions. Then the characters in the resulting image are approximately segmented. Next the characters are classified and the syntax of the results of the classification is checked.

One could also imagine an extra stage for image enhancement, but actually all stages except the syntax check stage can contain some sort of image enhancement or image normalisation. There can be an enhancement stage for the whole image, one for the whole plate and one for the individual characters. The enhancement steps are assumed to be part of the corresponding stages themselves.

1.2 The assignment

The assignment was as follows:

> Design and implement a license plate recognition system. The starting point was an image with the corner points of the license plate indicated approximately. All stages missing should be designed and implemented as good as possible within the six months available for the graduation project. Special attention should be paid to eliminating misclassification. A thorough literature study is needed to get insight into license plate recognition systems. The system should work for Dutch license plates, but should be designed in a way that it can be adapted for foreign plates without much work. The permitted time to recognise one plate is unknown and should not be of primary concern, especially because the system is perhaps implemented on custom hardware later. This usually causes the processing times to be reduced tremendously.

1.3 The application areas

A justified question at this time is: "What do you want to do with a license plate recognition system?". This is unknown at the moment. One can suspect that it could be used to bill speeders because of the involvement of the government. However, there are many other areas where a license plate recognition system can be useful. It could be used to:

- study travel habits. This could help solving the traffic jam problem in Holland;
- operate a no-stop tolling system;
- operate a ticket-free parking lot;
- secure gas stations;
- tell drivers that they are exceeding the maximum speed(with signs next to the road).

There are at the moment some experiments in the Netherlands with signs next to the road which indicate if the car passing is going too fast. These signs will have a much higher impact if the sign also mentions the license plate number of the car. It is important to reduce speeding because this has been found to be causing traffic jams. In general one can say that a license plate recognition system is beneficial for the safety and traffic flow on the roads.

1.4 The approach

It is always a difficult task to come up with a good approach. Obviously a literature study is a good starting point. A bottom up approach seems to be the most optimum for the project. The problems that could occur in a license plate recognition system are very hard to predict beforehand. With a bottom up approach it is possible to get a complete system running in a short time and then study the performance of the different stages. Then the bottleneck of the system should become clear and can be tackled.
It was decided to use Matlab to implement the programs that are needed. Matlab programs usually run very slowly and the programming language of Matlab is not well defined, but for this project it is very well suited. With Matlab a gigantic amount of useful functions for statistical and image manipulating are available. This is ideal for rapid prototyping, which is very important for this project. At the end the methods used are important, not the implementation. However, before anything is implemented, a literature study is done, which is described in the next chapter.

2. The literature study

An extensive literature study was done at the beginning of the project described in this paper. It should give insight in the problems that a license plate recognition system faces. The study should yield the following information:

- Which classification methods are suitable, and what are the benefits and drawbacks of each method?
- What are the bottlenecks of already existing systems?
- What is the performance these systems?
- Are license plate recognition systems still developed nowadays?
- What can image enhancement do to improve a system?
- Which character segmentation schemes are applied?

A separate report was written in Dutch about the literature study. It is available at the faculty together with many articles related to license plate recognition systems. This report was written for the so-called "Bibliotheekpracticum", which is a course on how to do and report a structured literature search.
The answers to the questions asked are roughly discussed here.

From the found literature it is concluded that most license plate recognition systems use neural nets to classify characters. These nets seem to be "smart" enough to deal with this difficult problem. The neural nets are most of the time used in combination with some kind of feature extraction like for example connected component detection. Systems applying neural nets have the best performance according to the articles. Some systems use template matching and some use special transformations. But then the system performance is less compared with the neural net approach.
Most systems described in the articles have the same problems. They have difficulties in classifying characters that look about the same, like for example the "**5**" and a "**S**" or the "**8**" and a "**B**". One can draw an important conclusion from this. All classification methods will most likely have weak points, this could be for example the classification of the "**5**" and the "**S**". Applying an extra stage specialised on a weak point probably offers a solution. This multi-stage classification approach was not found in any of the studied articles but has probably much potential.
It is often hard to judge the performance of a system from a description in an article. The performance is sometimes mentioned, but the conditions that lead to the stated performance are usually not, at least not in detail. In most cases the rejection rate for the whole system is mentioned, but the rejection rate per stage is not. The only conclusion one can draw about the performance is that the plate rejection rate of the whole system can be high, like 25 percent, if a very low, like 0.1 percent, misclassification rate is needed [XI]. The license plate segmentation is claimed to work almost flawlessly in some systems.

From the amount of recent publications it is concluded that much research is done on license plate recognition systems nowadays. A lot of research is done on electronic recognition systems with radio transmitters. However, an optical recognition system is always needed as a backup, so license plate recognition systems are still examined.
There was not much information in the articles about license plate enhancement and segmentation. Many enhancement techniques only improve images for the human spectator. An enhancement is sometimes of no use for a classifier, it can even deteriorate the performance of a system.
About image enhancement the articles only state that thresholding a license plate image without losing relevant information is very difficult if not impossible.
Classification methods will be examined in more detail in the next chapter.

3. Classification methods

Many different classification methods are discussed in literature. All methods found can be placed in three main groups, sorted by the strategy they apply. The three basic classification methods are template matching, artificial neural nets and transformation methods like the Hough transformation. The three classification methods will be discussed and compared with each other in the context of license plate recognition. Each group has many subclasses, only some of them will be dealt with here.
All methods have their own advantages and drawbacks. It is important to understand the problems that a license plate recognition system faces if one wants to judge each method. A list of properties of license plate recognition was made to get an idea about these problems:

- High noise levels
- Much distortion
- Many different circumstances
- Low optical resolution
- Low contrast
- Just 1 font
- Only small variations in character size
- Known syntax
- Known character spacing

The first five properties are disadvantages, the last four are advantages of license plate recognition compared to optical character recognition, OCR, on scanned printed text.
There is another aspect that plays an essential role in license plate recognition systems, namely the elimination of misclassification. A system should be able to classify, but it also should give a measure that indicates how certain the system is about a classification. With this measure one can decide to accept or reject results and so influence the rejection and misclassification ratio. A balance between the rejection and the misclassification has to be found. There are no boundaries given for the rejection or misclassification rate, but a system with a high misclassification rate is obviously useless.
The three main classification methods will be discussed in the next paragraphs.

3.1 Template matching

Template matchers are basically comparing characters with prototypes. The prototype that matches best to the character "wins". There are many ways to calculate matching costs. One way is to multiply the prototype pixel values with the corresponding pixel values of the unknown character and sum the results. The pixel values can also be subtracted from each other and then squared and summed. This is how template matching is implemented most of

the time. A special way of calculating the matching costs is applied in elastic template matching. Here the costs are a measure of the morphing energy that is needed to transform a character into a prototype.
If one looks at the list of properties in the previous paragraph then template matching does not seem to be a bad method. Especially because only one font with a limited variation in character size has to be recognised. Template matchers perform very well if used to classify characters that resemble the prototypes very well, which partially seems to be the case here. An important question is: "Are the matching costs a good measure to classify characters?". The best way to find out is to do some experiments. This can be done in a short amount of time because an elastic template matcher is already available and building a template matcher is fairly easy. Doing some experiments will also increase the understanding of the problem of license plate recognition. At this time no experiments have been done for this project with any kind of template matcher.
An important drawback of template matching will most likely make this method unsuitable for license plate recognition. The next example points out this drawback. Assume that the characters in figure 3 are the prototypes of a template matcher.

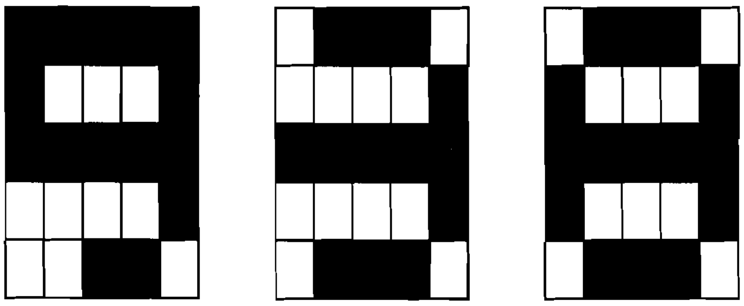


Figure 3. A prototype "9", "3" and "8".

In addition, assume that the character in figure 4 should be classified.

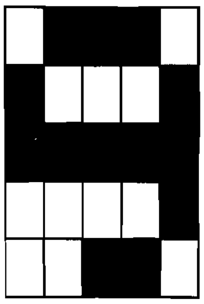


Figure 4. A sample character.

If the differences in grey level of each square of the sample character and the prototypes are summed for each prototype then the result will be two for each case. This yields that the sample character has to be rejected because there is no winner. This *is* probably the correct thing to do but that is not the point here. The "distance" between the prototypes and the sample character does not seem to be equal for all prototypes if one just looks at the example. This is a somewhat subjective measure, which is caused by the way humans identify characters. Humans pay more attention to structures like loops and strokes than to pixel data. The example shows that all pixels are treated equally which is far from optimal. Not all pixels are equally important. Their location and the values of all other pixels in the image specify how important they are. The next examples will go more into this drawback of template matching for license plate recognition. Figure 5 shows two manually distorted license plates.

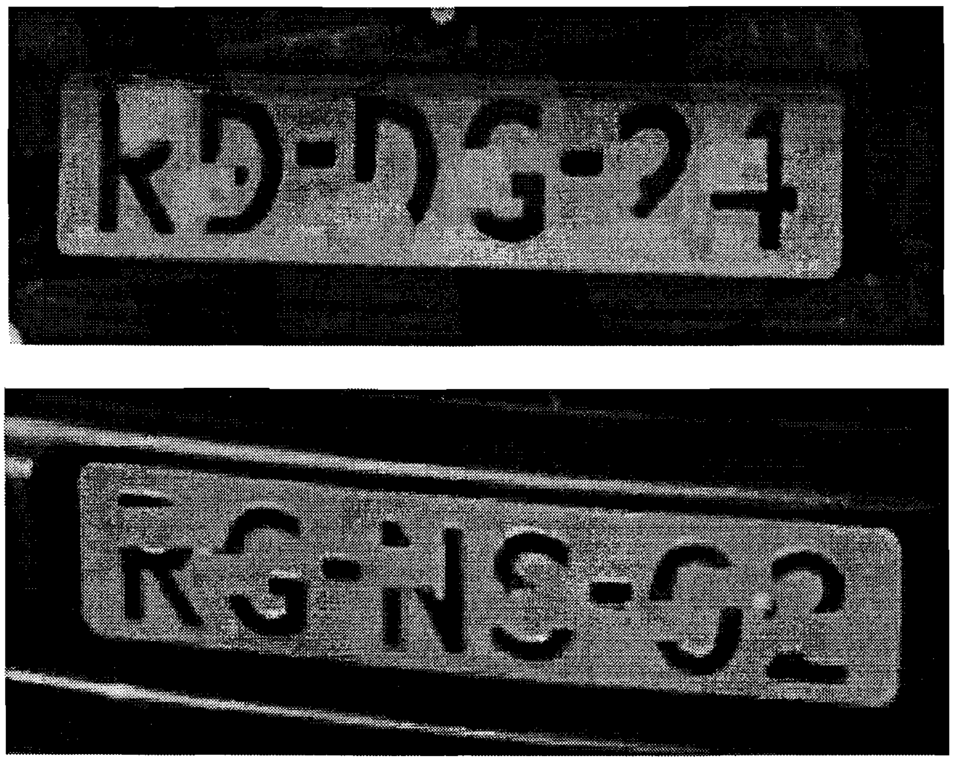


Figure 5. Two manually distorted Dutch license plates.

Most people can classify the plates in figure 5 without having any doubt, in spite of the heavy distortions that were applied manually. This is remarkable because humans are able to do this even without knowing what exactly the prototypes look like. The distortions in the plates apparently cannot confuse humans. The parts of the characters in the plates that are not removed give enough clues about the identity of the characters. Template matchers usually treat all pixels as equals. Changes in pixel values, if compared with the prototypes, will all contribute the same to the matching cost, independent of the position and the significance of the pixel. Obviously all pixels forming a character are not equally important to recognise the character as is shown in this example. Template matchers are not aware of the specific features that distinguish one character from the other. They just look at the differences between a character and the prototypes. This is why this approach is probably unsuitable for license plate recognition systems. A more advanced method is needed. Literature claims that neural nets offer a solution, therefore neural nets will be examined in the next paragraph.

3.2 Artificial neural nets

Artificial neural nets can be useful in a very wide range of problems. They are very suitable to model systems that are hard to describe mathematically. In image and character recognition neural nets proved to be especially useful. Neural nets seem to be able to cope with these kinds of problems.
A neural net is capable of "learning" the behaviour of a system by "looking" at sample data, presented in the form of vectors, of the system. A system is treaded as a black box, only the response to a number of inputs is known. The net "learns" by training it. This is done by presenting sample data to the net with the help of a training algorithm. After training it usually can accurately predict the response of even very complex systems to input data that is never saw before.
The set of samples used to train the net is called the training set. A test set is used to check the performance of the net. Sometimes this test set is used as a stop criterion for the training. In this case also a validation set is needed to measure the performance of the net independently of the training and test set. Usually all sets contain about the same amount of samples. A lot of sample data is needed for training and testing. This is considered a drawback of neural nets. The test and training set should be a good representation of the real life situation. Special attention should be paid to this. After training it is hard to find out which criteria the net uses to draw conclusions. Therefore, a bad test and training set will not immediately betray itself after training.
From the found literature it is concluded that applying neural nets is the most popular approach for license plate recognition systems. Applying them can be done in many ways. The grey levels can be used as inputs for the net immediately or some kind of feature extraction like for example connected component detection can be used. If the grey levels are fed to the net then a large net is needed, which implies that a lot of sample data must be available. As a rule of thumb there must be about five times as many samples in the training set as the number of weights in the net.
Many net structures are possible. The dimensions and thus the degrees of freedom in a net should fit to the problem that it is facing. It should have enough degrees of freedom to solve the problem. But if the net has too many degrees of freedom then the net starts paying attention to irrelevant information like noise and is not able to generalise the problem, which will cause degradation of the performance. Finding the right net structure for a problem can be a very difficult task.
A very nice property of neural nets is their flexibility. If after some experiment it turns out that the net has certain problems, like for example the classification of the "8" and the "B" then extra information can be fed to the net in order to train it for a better discrimination between these two classes. The application area of neural nets is not limited to classification

problems. A neural net can even be used to draw conclusions about the results of two independent classifiers or for image enhancement or segmentation.
A very common net structure is presented in figure 6. This net structure is called the feed-forward multi-layer perceptron architecture.

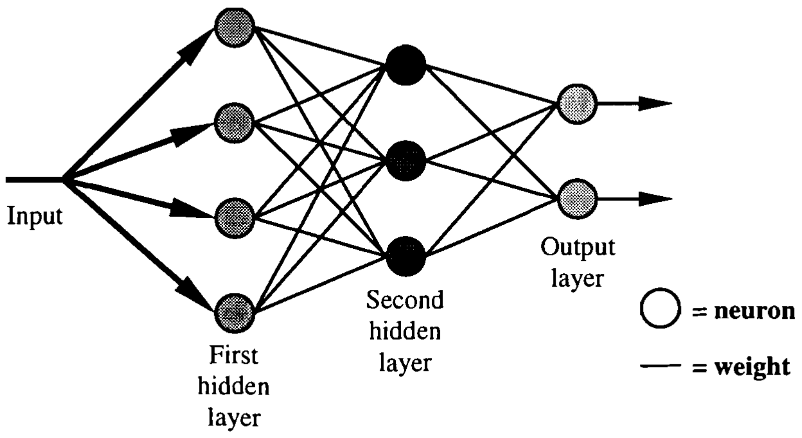


Figure 6. A multi-layer perceptron architecture.

The neural net in figure 6 has four layers, which are connected by weights. These weights transport data coming from the output of a neuron to the input of a neuron in the next layer. The neurons represent a non-linear function like a radial basis or a sigmoid function. This function describes the receptive field of the neuron. Often a Gaussian function defined by equation (1) is used for this.

$$f(D) = e^{-\frac{D^2}{2\sigma^2}} \tag{1}$$

With D the distance from the centre of the radial basis to a point, and σ a parameter describing the spread of the function. Figure 7 shows two radial basis functions with different σ values.

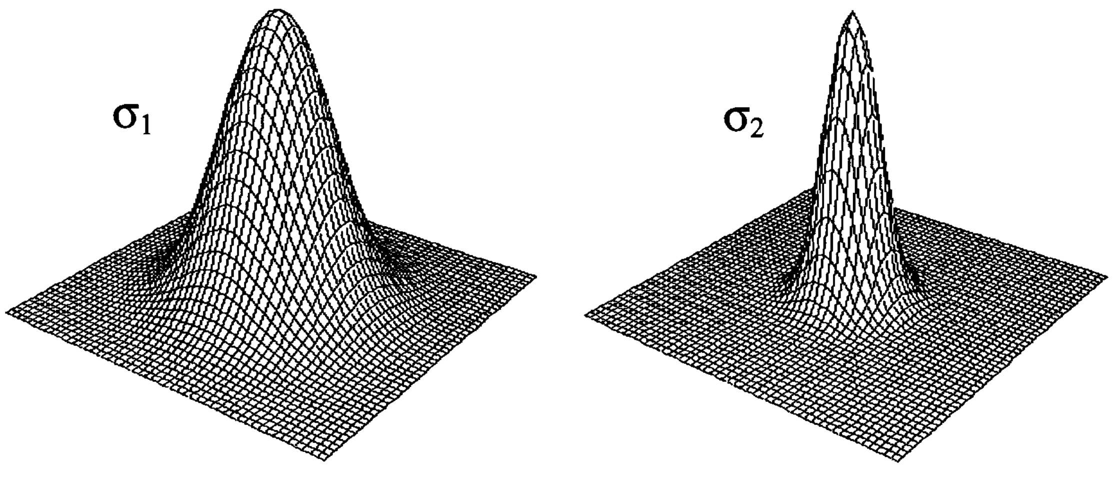


Figure 7. Two radial basis functions, $\sigma_1 > \sigma_2$.

The distance can be calculated in many ways. It does not even have to be measured equally in all directions. This can also be modelled with different σ values for each direction. For the 2-dimensinal case this will result in equation (2).

$$f(D(x,y)) = e^{-\frac{D_x^2}{2\sigma_x^2} - \frac{D_y^2}{2\sigma_y^2}} \qquad (2)$$

The receptive field of the neuron can then look like figure 8.

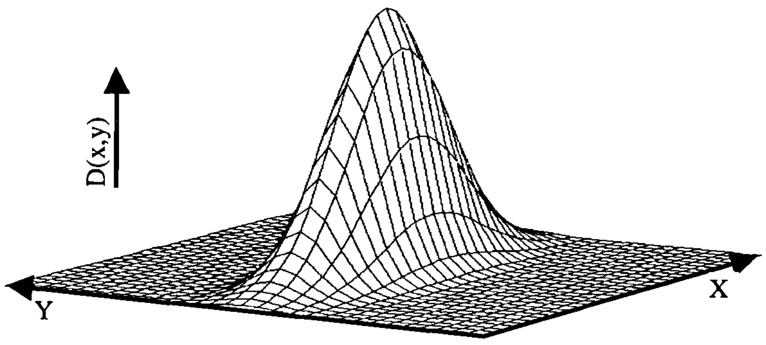


Figure 8. A special radial basis function.

How to train a multi-layer perceptron is described in the next paragraph.

3.2.1 Training a neural net

Many different training algorithms can be used to train a neural net. Training is most often focussing on reducing the mean square error of the output vector in relation to the wanted output vector. In most cases only the weights in the net are adapted to achieve this. This can be a very complex task, depending on the network structure. An imaginary mean square error function is plotted for just two weight values, w_1 and w_2, in figure 9.

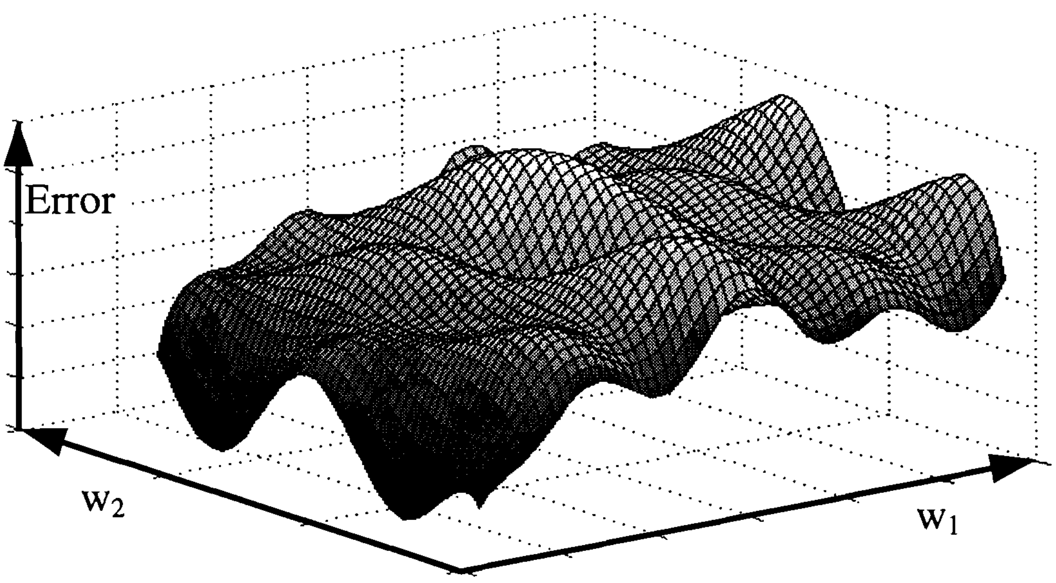


Figure 9. An error function.

Finding the global minimum of the mean square error function is usually the goal of training. The error function can contain local minima. It is difficult to tell if the result of training is the global or a local minimum.
A very popular training method is the back-propagation training algorithm. How this method works is illustrated in figure 10.

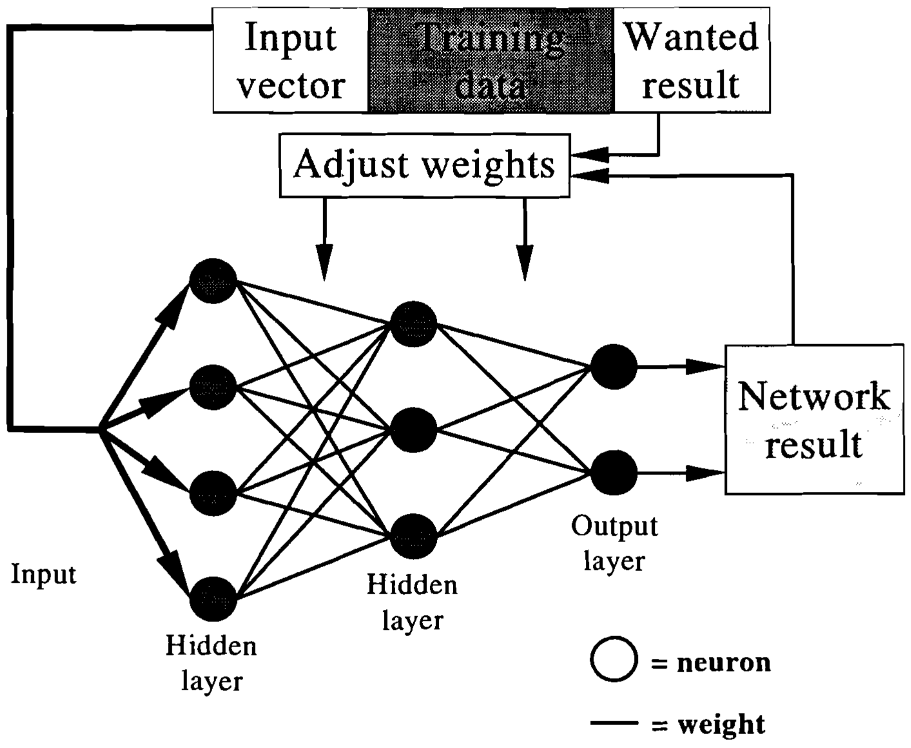


Figure 10. Back-propagation.

The weights in the net are initialised before training it. They can be initialised at random, but for the training speed and stability it is better to find an initialisation that is as close as possible to the expected weights after training. Training can cause oscillation of the weights. This is called instability of the training. If this occurs then the weights do not converge to a steady state. An initialisation for the weights can be found by solving the equations in the net for a number of training vectors. In some cases an initialisation can be *calculated* that is equal to the optimum solution in terms of the mean square error of the net. An example of such a net is shown in figure 11.

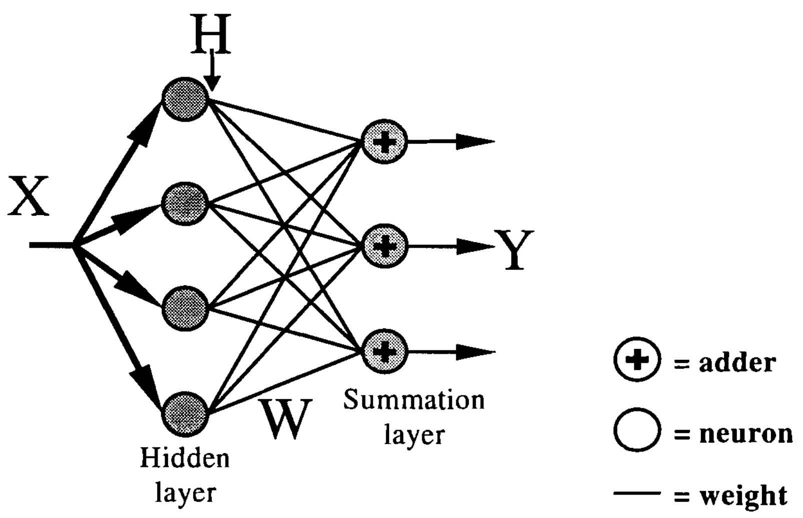


Figure 11. A dual-layer network.

The summation layer does not consist of neurons but of adders. The outputs(H) of the input can be calculated for all training vectors(X). Then the output of the net(Y) can be calculated with the weights(*W*). In matrix form this yields equation (3).

$$Y = W.H \tag{3}$$

The wanted outputs for each input vector of *X* are known, these are the target vectors *T*. For the optimal weight matrix this yields equation (4).

$$T = W_{opt}H \tag{4}$$

Thus the optimal weight matrix is defined by:

$$W_{opt} = T.H^{-1} \tag{5}$$

So W_{opt} can be found by calculating the inverse of matrix H. If H is not a square matrix then there is no inverse of H. In this case the pseudo inverse can be used. Equation (6) gives the definition of the pseudo inverse.

$$H^{Pseudo} = H^T((H.H^T)^{-1})^T \tag{6}$$

Thus the optimal weight matrix is defined by:

$$W_{opt} = T.H^{Pseudo} \tag{7}$$

So in this case only training data is needed to find an optimum weight matrix, but in most cases training the net is needed.
The weights are adapted in the training stadium. This can be done for each training vector separately or for the whole training batch at once. Batch training is more stable and quicker to calculate. There are many strategies to adapt the weights. To examine if the result is a global or a local minimum one can try different initialisations of the weights before training. If the outcome is the same every time then the result is probably the global minimum.
The network in figure 11 can be trained by adapting the weights with ΔW of equation (8).

$$\Delta W = L_R(T - Y).H^T \tag{8}$$

With L_R the learning rate and T the target matrix which defines the wanted output for each training vector. The response of the net to the training vectors is represented in Y. The learning rate defines how much the net is updated each time. An appropriate value must be found. If the value is small then the training can take very long. The net can start oscillating if the learning rate is too large.
The definition of the error at the output of the net depends on the problem that it is trying to solve. If one for example is only interested in the highest and second highest response at the output layer then the error of the net is not defined by the mean square error. Let us assume that one wants the highest response to be 1 and the second highest response to be 0. A target vector with zeros and one 1 can be used for this. This way one "pushes" all outputs except one to go to 0. Which is more than one needs. It is better to only adapt the weights that are related to the output that one wants to be 1 and the highest other output. This can be achieved by resetting the outputs that are of no interest to the wanted output levels. Thus not all weights are adapted. This leads to a better performance of the network.
Not all neural nets require training, like for example a Bayesian classifier implemented as a neural net. This is an example of a probabilistic neural net. These nets are described in the next paragraph.

3.2.2 Probabilistic neural nets

The Bayesian classifier can be implemented as a probabilistic neural net [II, chapter 3] [IX chapter 3]. A Bayesian classifier is ideal to classify point clouds. Each training sample is stored in a neuron in this approach. A Gaussian radial basis function can be used as the receptive field of the neurons. The response of all neurons of a class is summed with an adder. The class associated with the adder with the highest result is assumed to represent the correct classification result. The more neurons in a class the higher the probability that the adders of the class "wins". A class can be interpreted as one neuron with a complex receptive field. The receptive field of this neuron is the sum of the receptive fields of all neurons of a class. These properties of the Bayesian classifier result in two very important advantages over other classifiers:

- No assumption is made about the shape of the distribution of probability, it solely depends on the distribution of the training data.
- It implicitly reckons with the a-priori probability of a class.

Equation (9) shows how to calculate the response $f_a(X)$ of a Bayesian classifier for class A.

$$f_a(X) = \sum_{i=1}^{n_a} e^{-\frac{(X-Y_{ai})^T(X-Y_{ai})}{2\sigma^2}} \tag{9}$$

with i the training vector number, σ the smoothing variable, n_a the number of training vectors in class A, X the test vector to be classified, Y_{ai} the ith training vector of class A and T the vector transpose. Figure 12 shows a Bayesian classifier implemented as a neural net.

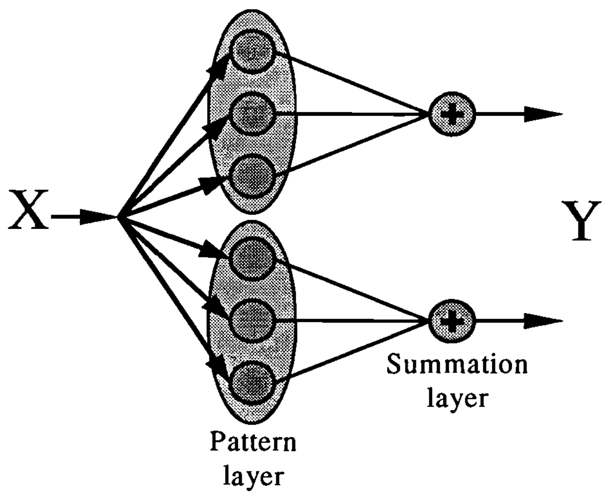


Figure 12. A neural net implementation of a Bayesian classifier.

No training is required for this kind of network. The spread of the non-linear function of the neurons that form the receptive field of a class must be set to an appropriate value. Let us assume that a Gaussian radial basis function is a the receptive field of the neurons, then a appropriate value for σ must be chosen [II, p. 3.10]. If the sigma value is small then almost only the neuron closest to the sample data will respond. The second closest neuron will respond much less. The classifier then almost works like a neighbour classifier. If the σ is large then the most probable classes will overwhelm the other classes. Figure 13 shows an example with two classes. The receptive fields are shown in one dimension. Class *B* has two times as many sample vectors as class *A*. The sigma in the right image is larger than the sigma of the left image. The response of class *A* on input x_i is higher than the response of class *B* in the left image. It is the other way around in the right image.

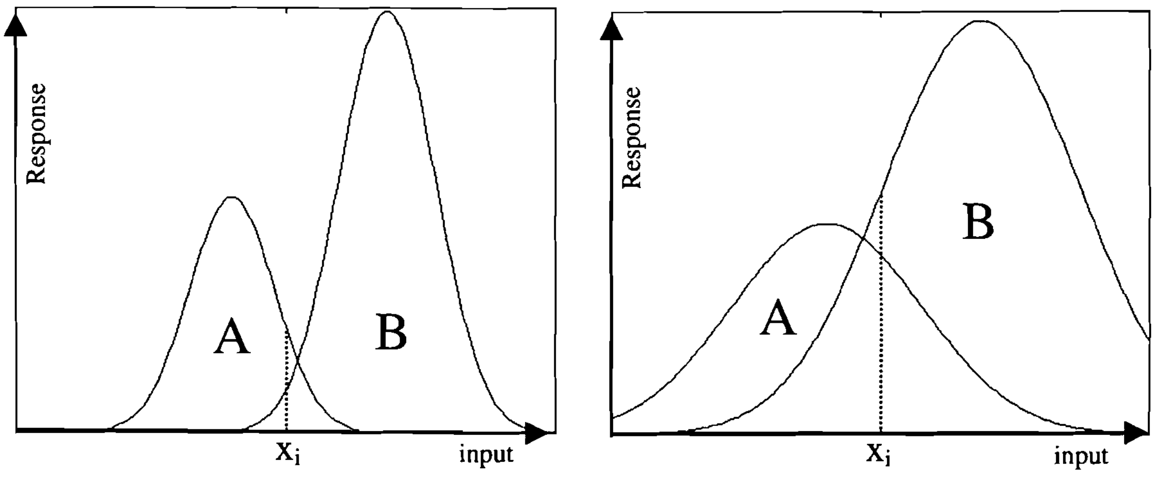


Figure 13. The receptive fields of two classes for different sigma's.

Figure 13 shows that the classification result depends on the chosen sigma value. Figure 14 shows an example of the receptive field of a class for two different sigma values.

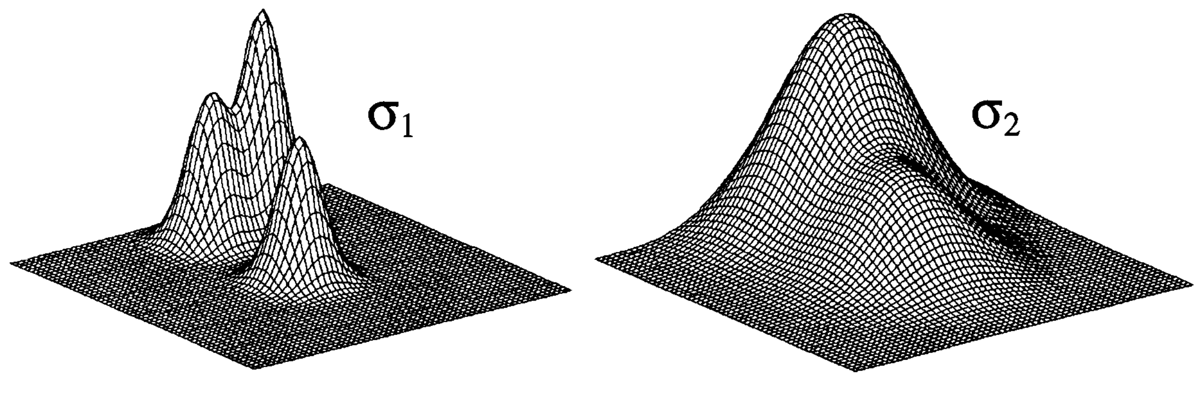


Figure 14. The receptive fields of a class for different sigma's, $\sigma_1 < \sigma_2$.

The chosen σ_1 in figure 14 is probably too small, σ_2 is chosen better.
If there are many neurons then this approach is very computational intensive. There are methods to reduce the number of neurons, and thus the computation time, without decreasing the performance of the classifier. One could for example approximate a cluster of neurons with one neuron with a large receptive field. This technique is called clustering [IX, p. 163-165].
An advantage of the Bayesian classifier is that no training is required. If enough good sample data is available then this approach results in an optimal classifier for point clouds. It is possible to test this type of classifier without an independent training and test set. Then all except one sample is used to "train" the net. For each test cycle only one sample is excluded from the train set. This sample is used to test the net. This way the test set is independent of the train set, but no separate sets are needed.

Neural networks seem to be promising. The possibilities should definitely be explored.
Many classifiers based on neural nets use some kind of feature extraction. This is done to reduce the dimensions of the net, which would be quite large is the pixel data was fed to the net immediately. A transformation can be used to classify or extract features or reduce the dimension of pixel data. Some transformation methods are described in the next paragraph.

3.3 Transformation methods

Many transformations on characters can be used to classify them, but this is not commonly seen in license plate recognition. Transformation methods can also be used for feature extraction or data reduction. This is often done in combination with a neural net. Two transformation methods will be discussed here. The first one is the Hough transformation.

3.3.1 The Hough transformation

The Hough transformation [VIII, p. 7.1-7.6] is based on contour information. It is suitable to find objects in an image that can be described with simple functions like a straight line or a circle. Some articles found in literature successfully use this transformation to segment license plates.
Accurately finding the contours of characters on license plates is very difficult because of the low resolutions used here. The different circumstances of the license plate images makes it even more difficult. That is why this transformation in not likely to perform well for license plate classification.
A transformation method that already was examined in a former project [I] is the PCA transformation. It is described in the next paragraph.

3.3.2 The PCA transformation

A student examined the possibilities of using the principal component analysis, PCA, for multi-font character recognition in a former project[I]. The results were very encouraging. The system showed that PCA is very suitable to reduce the dimension of data without losing relevant information. This is ideal in combination with a neural net because then dimensions of the net can be reduced and thus only a small training and test set is needed. The classifier proved to be very noise resistant. In this system the pixel data was PCA transformed. The result was a vector with a much lower dimension than the pixel data. The PCA transformed character was then classified with a radial basis neural network. The scheme of the system is presented in figure 15.

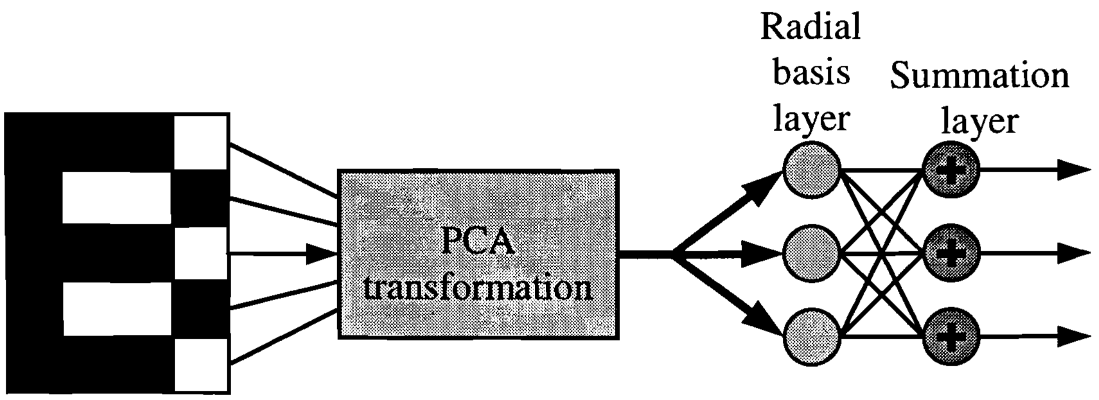


Figure 15. Classification with PCA transformation and a radial basis neural net.

The PCA transformation seems to be able to reduce the dimension of the pixel data without losing relevant information. If there are *N* prototypes then the PCA transformation of a prototype has at most *N*-1 elements. This number is independent of the number of pixels in the prototypes. The prototypes can even be reconstructed exactly from their PCA transformation. Only a small neural net is needed to build a classifier if the principal component analysis is used, so the amount of train and test samples required is much less than without the transformation.

This seems to be a very promising approach for license plate recognition also. It will be examined in more detail in chapter 8.

4. Dutch license plates

All motorised vehicles on the Dutch roads must be registered at the ministry for traffic regulations. They assign a license plate number to each vehicle. The number is placed on a license plate, which has to be mounted on the vehicle. A plate is usually mounted symmetrical to the bumper of the vehicle, but placing it more to the sides is also allowed. All vehicles except motor cycles have plates at the back and the front. Motor cycles only have plates at the back.
The legal requirements for Dutch license plates allow about 60 different models. These differ from each other in height, width, colour, text font or layout of the characters. The different shapes of plates can directly influence the license plate and character segmentation stage. The fonts used in the plates can directly influence the classification.
The regulations concerning the different models are very strict. This can help in segmenting and recognising license plates.
The next paragraphs only describe the most common Dutch license plates models.

4.1 The common Dutch license plate

Figure 16 shows an example of the most common Dutch license plate model.

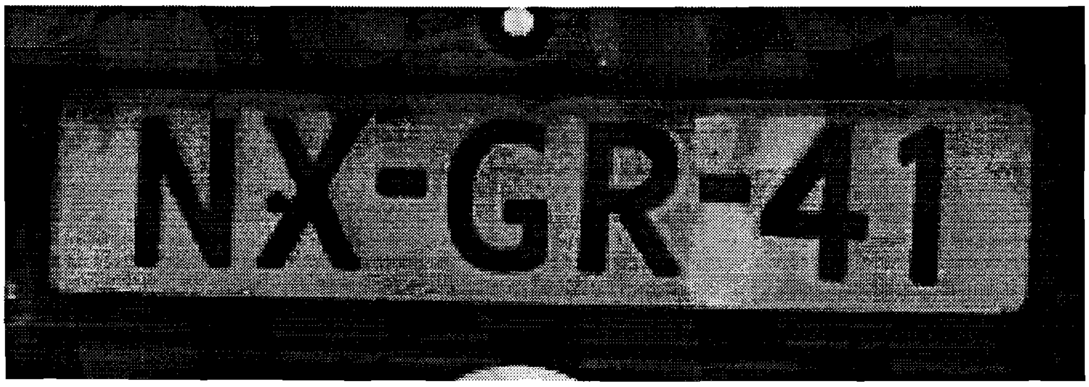


Figure 16. The most common Dutch license plate model.

The background of the common Dutch plate is yellow, the characters are black. Almost every Dutch license plate is of this type. That is why was chosen to only focus at this type of plate at first. At a later stage the developed character segmentation and classification can be expanded for other types. The exact percentage of the Dutch plates which are of this model is unknown, it is probably about 98 percent.
The plate height may be 11 or 12 cm, usually it is 11 cm. The license plate segmentation stage should not have any problems with these differences. The height of the characters is

proportional to the plate height. Therefore, if the plates are scaled to a fixed size then the character dimensions should be the same. The character height is about 8 cm if the plate is 11 cm high. The plate is 52 cm wide.
The characters are usually punched in a metal plate, so they have some depth. There are also variants where the characters are glued on the plate. These characters are about 1 cm thick. This can cause some shadow around the characters on the plate.
Only a few fonts are allowed in license plates. Law regulates the character dimensions and spacing. In practice only one font is seen in the most common type of plate. If one only focuses on this type of plate then one has to deal with just one font.
The characters must be placed symmetrically in a plate. Therefore, the space above the characters must be the same as the space below them because all characters have about the same height. The space to the left of the first character and the space to the right of the last one must also be the same.
The contents of a plate has to obey strict syntax rules. It consists of three pairs of characters separated by two dashes. A pair always consists of two digits or two letters of the alphabet. Not all letters of the alphabet are used. The contents of a plate gives information about the type and age of a vehicle. Not all characters are used and some combinations of characters in a pair are not allowed.
A syntax check stage can be used to check if the recognition results are according the regulations and thus detect some misclassifications. If for example an "8" is classified as a "B" then the syntax check stage will detect that the syntax is invalid.
Another thing that can be checked after classification is the character spacing. If a misclassification is caused by bad segmentation then the character spacing found is probably invalid. Checking this can help in reducing misclassification.
A new model will replace this type of plate in the near future. The new model is outlined with a thin black line. This is probably beneficial to the license plate segmentation stage. The syntax of the contents and the plate dimensions are the same as the common model. A blue area is added in the left part of the plate. The blue area and the outline are the cause that the characters are a little smaller than in the common model. The symbol of United Europe, a circle of stars, is placed in the blue area together with a flag and an abbreviation of a country. This type of plate is called the Euro plate, it has not been released yet so no examples are available. A sticker is sometimes put on new plates to make them look like a European plate. This is actually illegal. Figure 17 shows an example.

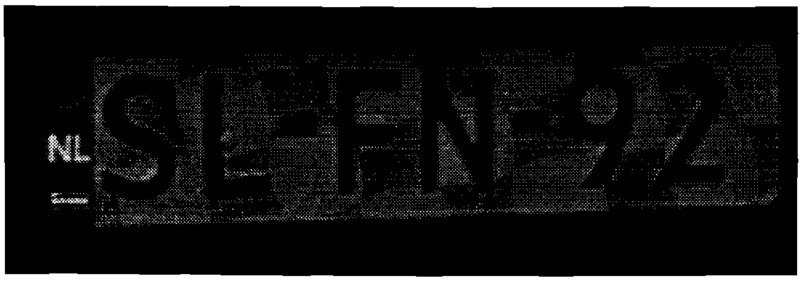


Figure 17. A fake Euro plate.

4.2 Other types of plate

Besides the most common Dutch license plate, described above, several other types can be distinguished. For instance figure 18 shows the so-called "square" plate. The plate is not square of course. These plates are mostly used on trucks and hatchbacks.

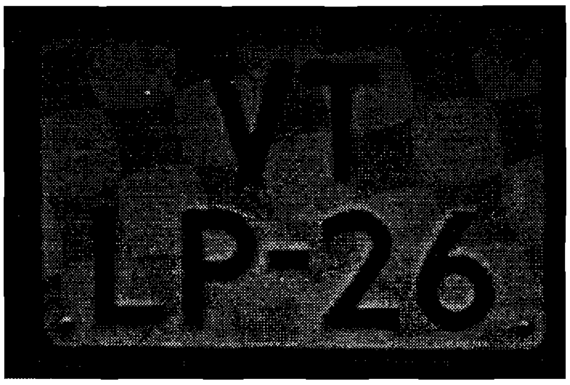


Figure 18. A "square" plate.

Motor cycles have much smaller plates than cars. An example is shown in figure 19. These plates usually contain a "M".

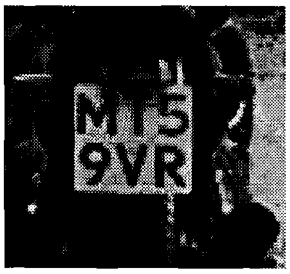


Figure 19. A motor cycle plate.

Another type of plate is presented in figure 20. Two bolts and an APK[2] sticker are also visible in this image. This plate is less wide than the common plate. This model is called the "American" plate because it is mostly seen on American cars, which have less space reserved for the license plate. A special permit is needed for these plates.

[2] A sticker that shows that an obliged annual inspection has been executed on the car, in Dutch APK stands for "Auto Periodieke Keuring", which means car periodical inspection.

Figure 20. An "American" plate.

All previously described plates have a yellow background with a black foreground. The plate in figure 21 is a garage plate. It has a green background and a white foreground. Garages use it for test-drives. Special license plate numbers are used for these plates. Note that there is an "A" in this plate.

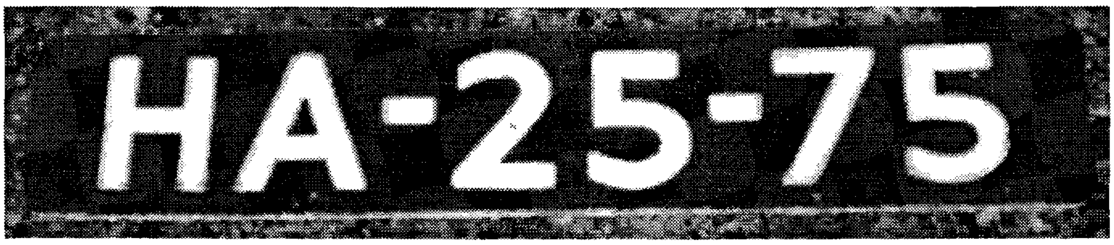


Figure 21. A "garage" plate.

The plate in figure 22 is very uncommon. It is a temporarily plate. This plate is just valid for a few days. It is used to import cars that do not have a license number yet or for trucks that are modified and thus need a new license plate number. This is the only plate that may be made by hand. It has a white background and black characters.

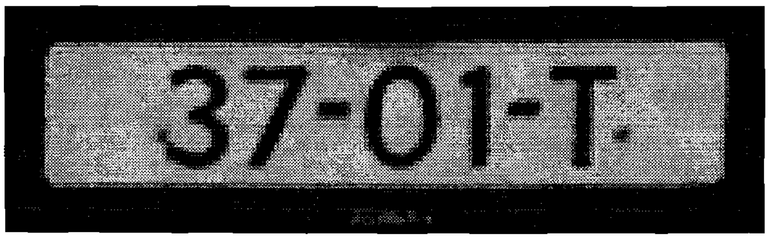


Figure 22. A temporarily plate.

Cars that got their license plate number before 1977 have plates with a blue background and a white foreground. An example is shown in figure 23.

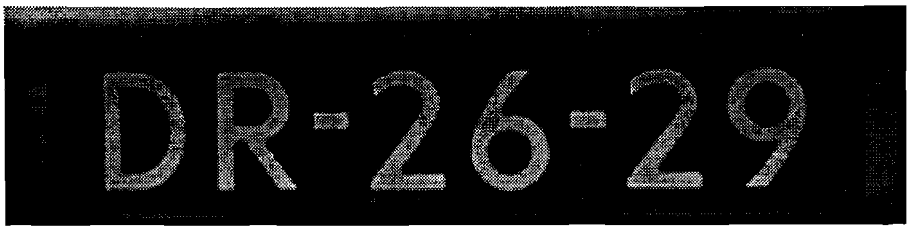


Figure 23. A plate of the old model.

Blue is the complementary colour of yellow, so inverting the colours of this image should result in an image that looks look like the common model, as is shown in figure 24.

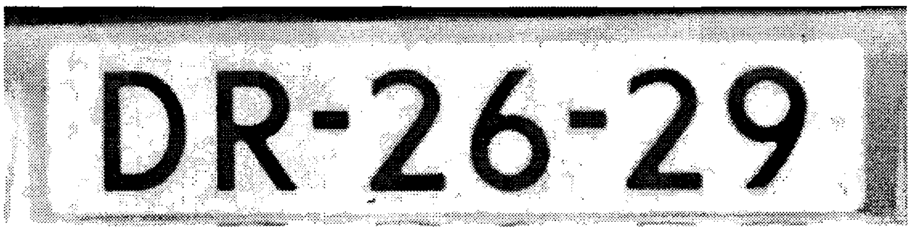


Figure 24. An inverted plate of the old model.

4.3 Conclusions and recommendations

The plates described in the last paragraph represent only a small amount of the total number of plates seen in Holland. The exact percentage is unknown. In a snap check of 1000 plates only a few plates were not of the common model. This makes it difficult to get a reasonable amount of sample images to experiment with these rare models. Therefore they are ignored at the moment. It is unknown how the license plate segmentation stage will react on these plates. It is best if the license plate segmentation stage detects what kind of plate is present in an image. In addition, foreign plates should be detected there. The aspect ratio and colour of a plate give clear clues about the type of plate. If the type of plate is known then the character enhancement, segmentation and classification can be adapted to it.

With all these types of plates is seems to be interesting to spend some time on examining the possibilities of segmentation-free classification. Perhaps an extension to the used segmentation error corrections discussed in chapter 9.2 can offer a solution.

5. The camera and frame grabber

The approach described here is designed and implemented by CFT, it is no part of the assignment. Not many details about this part of the system are known to us.
A high-speed shutter PAL[3] camera and a frame grabber form the front-end of the system. The camera delivers images of automobiles passing by. A frame grabber then digitises the images. Artificial lighting is used to assure a minimum level of illumination of the license plate. The angle between the road and the line between the camera and the license plate is assumed to be about 31°, like shown in figure 25. Some experiments done by the Dutch ministry for traffic regulations proved that this is the optimal angle.

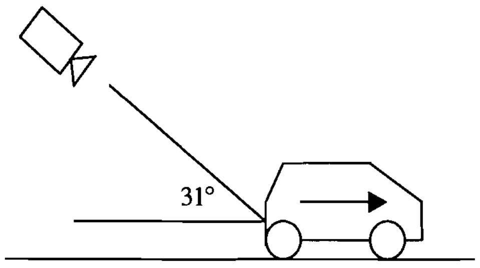


Figure 25. The camera position.

No trigger mechanism is used to start the camera or frame grabber, so a continuous flow of images is delivered. These images are evaluated by computer to find out if there is a license plate in the image. Examining all images is probably very time consuming. Most systems described in literature use trigger mechanisms like optical bridges or electromagnetic loops in the road. Which approach is best depends on the available computing power. The method of CFT will deliver the best results because it is independent of the vehicle length and speed. In addition, the placement and angle of the camera in relation to the road is less critical this way. More images of one plate are taken, which can be beneficial to segment the plate.
It is best to take images of the back of a vehicle instead of the front. Here are some reasons:

- Motorcycles only have license plates at the back.
- The lights at the back of a vehicle do not influence the images as much as the front lights.
- Many trucks have signs at the front, which can be mistaken for license plates.
- Drivers who want to avoid being seen by a camera can drive very close to the vehicle in front of them. The license plate will then be obscured.
- The plate at the back is usually cleaner than the front plate.

Some examples illustrating this are presented in figure 26.

[3] Phase Alternating Line

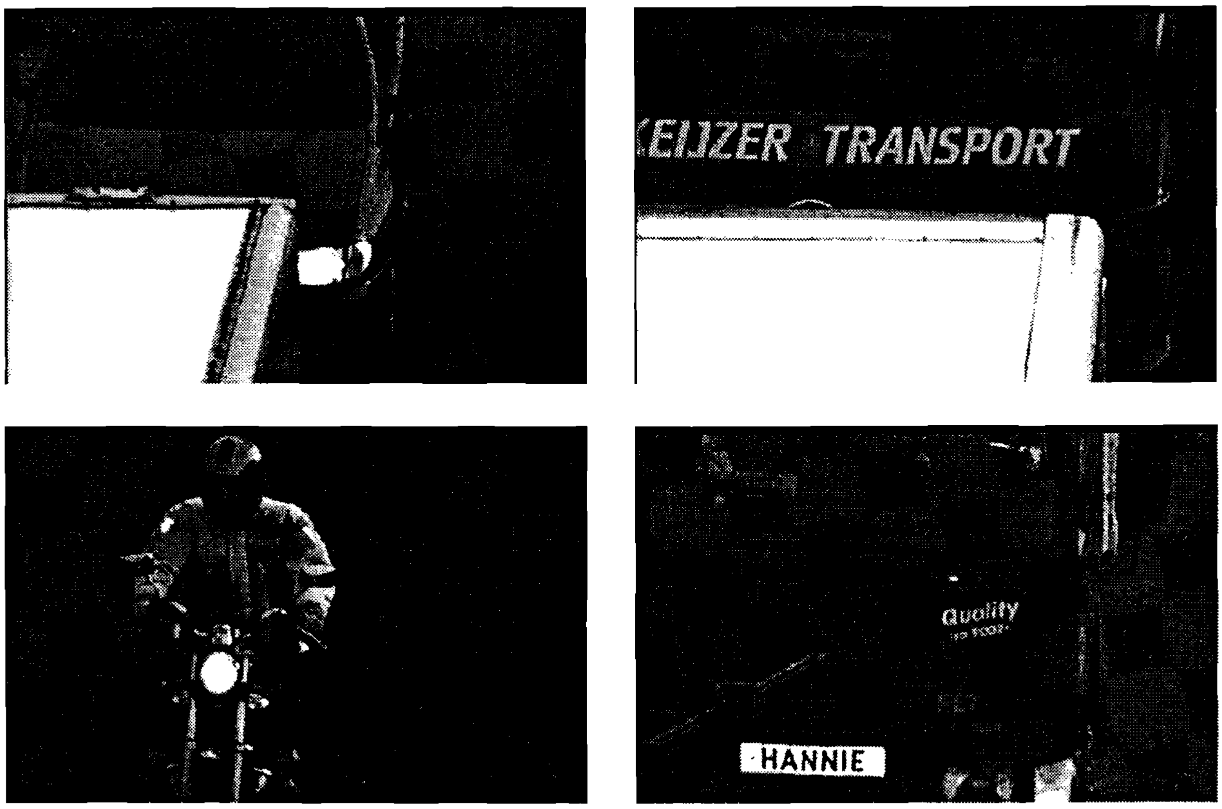


Figure 26. Images of the front of some vehicles.

These images were made on a highway. The vehicles drive about 100 kilometres per hour.
A disadvantage of taking images of the backside is that many Dutch cars have APK stickers on the plate at the back. These stickers have a lighter colour than the background of the common Dutch plate. This can lead to problems in the segmentation and recognition of the characters on the plate. The stickers now have become obsolete.
Usually more than one camera will be used, especially if there are more lanes in the road. A wide camera angle or more cameras must be used to make sure that no car can pass by without getting its license plate on camera. If a wide camera angle is used then the resolution of the license plate characters will be very poor.
The cameras use the PAL system. A PAL image consists of two frames, one containing the odd and one containing the even image lines. In total there are 585 visible image lines. Each line contains about 680 pixels. The Kell-effect[4][VIII, p. 2.6] limits the resolution of 585 x 680 to 439 x 510 pixels. The characters on a plate are expected to be at least 15 pixels high.
A special monochrome camera is used to sample two frames at once, this is necessary because the cars could be driving very fast. Usually every fiftieth second one frame is sampled which leads to 25 full images per second. If every fiftieth second one frame would be sampled then the two frames would be unsuitable to be used as one image in this application. A gain of a factor two in the vertical resolution is achieved by using this special camera.

[4] Kell factor 0,75.

At this time a monochrome camera is used, later a colour camera could be applied, which could help in recognising foreign plates. In addition, the license plate segmentation stage can benefit from a colour camera.
The license plate images coming from the frame grabber can look very differently. Partially this is caused by the artificial lighting. Dirty plates do not reflect much light, so they still stay very dark. Clean plates reflect a lot of light, which makes the background of a plate very bright. The characters then become very thin, almost skeletonised. The angle that the plate is facing in relation to the lights also influences the images. Figure 27 and 28 are examples of images coming from the frame grabber. These and actually all images used here are achieved a little differently than described above. A VHS[5] tape, recorded in PAL, was used to store the images of the camera. Later the images were digitised. This extra step reduces the optical resolution of the images and introduces some noise. The performance of the system will probably increase if this step is omitted.
More than 1000 images were obtained in two setups. The first setup contained 593 plates. 29 images contained foreign plates. 19 images contained totally unreadable plates, which are probably Belgian plates. The remaining 545 images were used to test the license plate recognition system. These 545 images were split up in two sets to be able to judge the performance of the system better. The images were split up in 461 good and 84 bad samples. The bad samples contain an obvious distortion in the license plate, which probably will cause problems for the character segmentation or classification. The images made in the second setup, where the image contrast and brightness was increased a little, can be used to develop the system further.

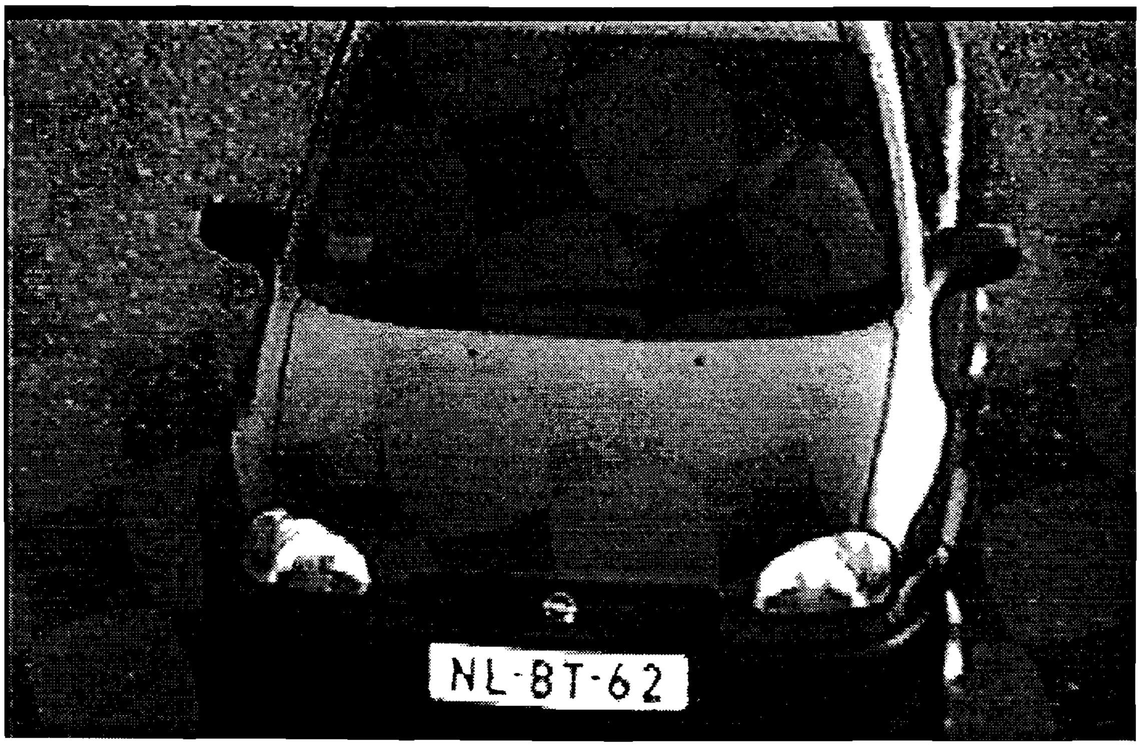


Figure 27. A car with a bright plate.

[5] Video Home System

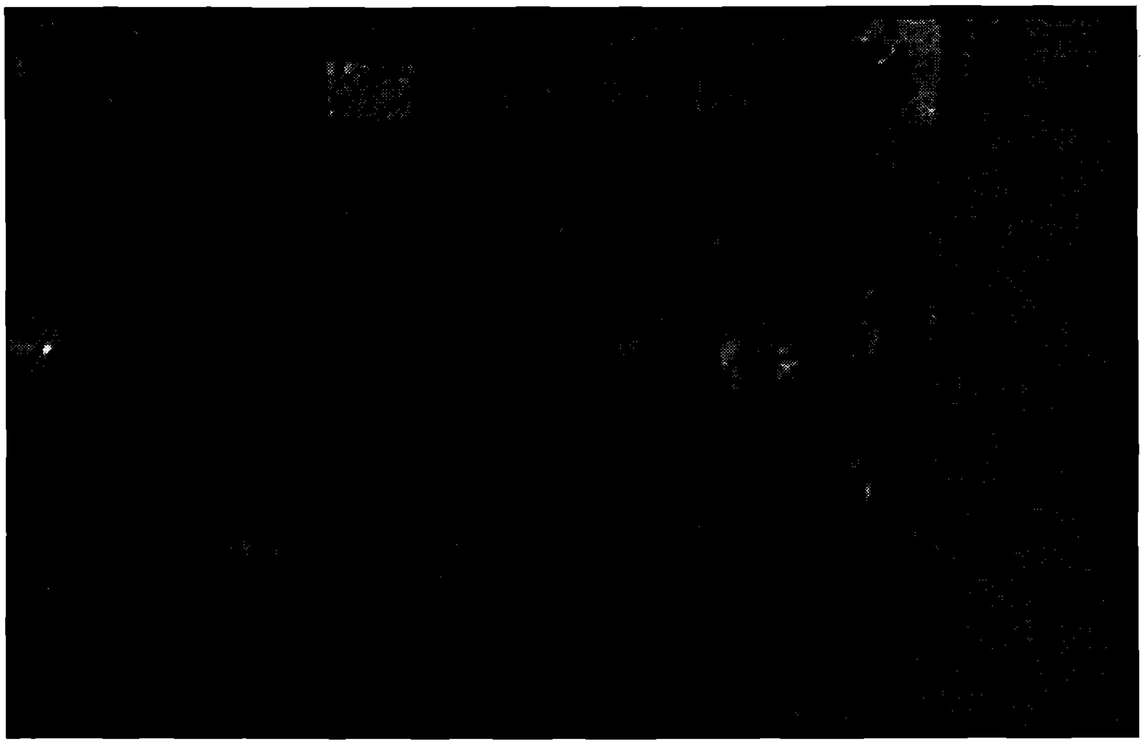


Figure 28. A dark image.

The images in figure 27 and 28 show that the results coming from the frame grabber can be very different. Some noise is clearly visible in figure 27, especially at the upper side of the image.
It is important to tune the setup of the camera and frame grabber correctly. The characters in bright plates should not disappear, and the background of dark plates should not become as dark as the characters.
A computer is continuously checking the images coming from the frame grabber for license plates. If an image is found with a plate in it, then the location of the plate has to be calculated. This is done in the license plate segmentation stage, which is described in the next chapter.

5.1 Conclusions and recommendations

Special attention must be paid to a good setup of the camera and frame grabber. Properties like contrast, sharpness, noise, camera angle and used artificial lighting all can influence the performance of the system.

At this time a monochrome camera is used. Using a colour camera could help in recognising foreign plates. In addition, the license plate segmentation stage can benefit from a colour camera.

6. License plate segmentation

The approximate positions of the four corner points of a plate are calculated in the license plate segmentation stage. CFT designed and implemented this stage for the most common plate. The license plate segmentation stage is unavailable to us at the time, so no experiments with this part of the system were done. From CFT some specification of the performance of the system are known. It is claimed that the system is able to segment the plates with an accuracy of about two pixels. These dimensions in pixels refer to the dimensions of an image of 439×510 pixels that is delivered by the frame grabber.
The rejection and error rate of the license plate segmentation stage are unknown. From literature it is known that a license plate segmentation stage can have a rejection rate as low as 1% and still have an error rate of almost zero. The details about the license plate segmentation are unavailable to us, only roughly the strategy is known. The license plate is found by applying template matching to find the corner points of the license plate. If four points are found that are possibly the corner points of a license plate then the contents of the quadrangle is checked on spatial frequencies. Certain spatial frequencies can be expected because of the characters in a plate. Template matching is applied to see if there are two dashes in the quadrangle. This implies that the approach is only suitable for Dutch plates.
An image with the four corner points of a license plate indicated is the starting point for the project described in this paper.
Due to the perspective view, these corner points may not correspond to a rectangle. The area described by the four corner points will be transformed into a rectangular area of fixed dimensions. This should correct for most of the perspective view distortion of the plate, but cannot correct for distortions caused by bent plates. The perspective view correction is done with the bilinear transformation, which is described in the next paragraph.

6.1 The bilinear transformation

The bilinear transformation can transform an area described by four points (*A*', *B*', *C*' and *D*') to a rectangular area(*A*, *B*, *C* and *D*), like shown in figure 29.

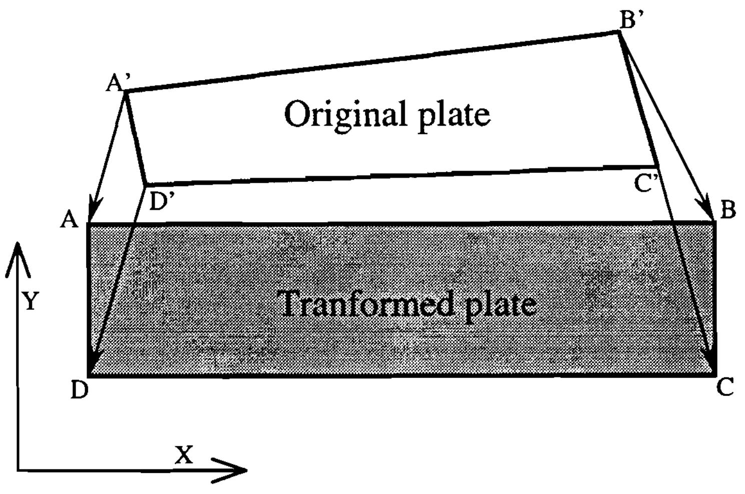


Figure 29. The bilinear transformation.

This transformation leads to eight equations, two for each corner point. For each corner point (*x*',*y*') of the original image the equations are:

$$\begin{aligned} x' &= \alpha_0 + \alpha_1 x + \alpha_2 y + \alpha_3 xy \\ y' &= \beta_0 + \beta_1 x + \beta_2 y + \beta_3 xy \end{aligned} \qquad (10)$$

with *x* and *y* defining the position of the point (*x*,*y*) in the transformed image. The 8 equations for the four corner points can be written in matrix shape:

$$\begin{bmatrix} 1 & x_A & y_A & x_A y_A \\ 1 & x_B & y_B & x_0 y_A \\ 1 & x_C & y_C & x_A y_A \\ 1 & x_D & y_D & x_A y_A \end{bmatrix} . \begin{bmatrix} \alpha_0 & \beta_0 \\ \alpha_1 & \beta_1 \\ \alpha_2 & \beta_2 \\ \alpha_3 & \beta_3 \end{bmatrix} = \begin{bmatrix} x'_A & y'_A \\ x'_B & y'_B \\ x'_C & y'_C \\ x'_D & y'_D \end{bmatrix} = M.[\alpha \quad \beta] = [X' \quad Y'] \qquad (11)$$

M contains the position of the four corner points of the transformed plate. These corner points can be chosen as needed. *X*' and *Y*' contain the position of the four approximately segmented license plate corners. The 8 unknown variables(α_i and β_i with $i \in \{0, 1, 2, 3\}$) can be solved with equation (12):

$$[\alpha \quad \beta] = M^{-1}[X' \quad Y'] \qquad (12)$$

The *α's β's* only have to be calculated once per transformation. Then one can calculate the positions of a point in the original image (*x*', *y*') for all points in the transformed image (x, y) with the help of equation (10).

There are two ways to interpret pixels. One is to interpret them as samples. Then the grey level of a pixel is defined by a sample that corresponds with the centre of the pixel. In the second interpretation, pixels are interpreted as areas. One interpretation can sometimes be translated to the other interpretation, as will be shown in this and the next paragraph.

With the help of the bilinear transformation one can calculate the areas in the original image that defines the grey level of each pixel in the transformed image. Figure 30 illustrates this.

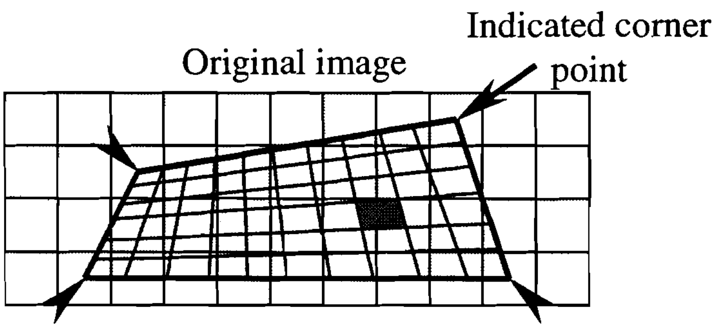


Figure 30. The areas defining the grey level of the transformed pixels.

So the small dark grey area in figure 30 defines the grey value of *one* pixel in the transformed image. The areas usually overlap more than one pixel of the original image.
To find the grey level of a transformed pixel one has to calculate how much of the area overlaps the pixels in the original image. This can be quite complex as one can see in figure 31.

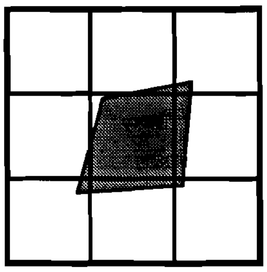


Figure 31. An area which overlaps eight pixels of the original image.

So eight pixels of the original image hold information about the grey level of one pixel in the transformed image in this example.
Exact calculation of all areas and overlaps that define the grey level of the pixels in the transformed image is very computational intensive and difficult to implement. An approximation of the area can be calculated with bilinear interpolation, which is described in the next paragraph.

6.2 Bilinear interpolation

In bilinear interpolation [VIII, p. 5.13-5.15] the grey level of an interpolated pixel is calculated, based on the grey levels of four pixels of the original image. For the centres of every pixel in the transformed image, the corresponding position in the original image is calculated, using the bilinear transformation. The grey level for this position(*P*) is then calculated based on bilinear interpolation of the four neighbouring pixels in the original image, which is illustrated in figure 32.

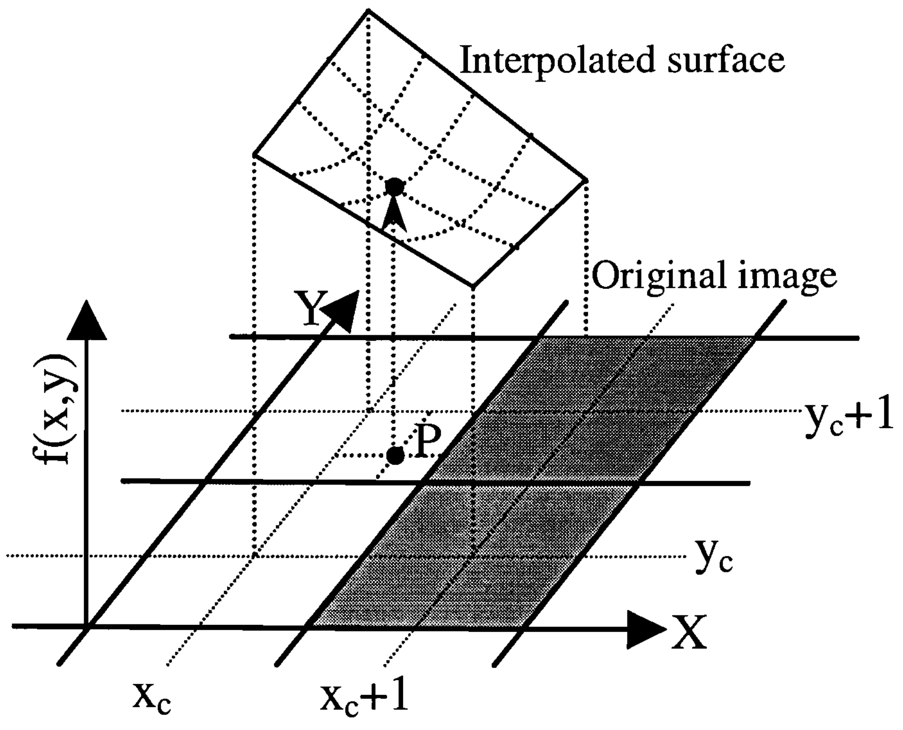


Figure 32. Bilinear interpolation.

The four corner points of the plane, which is a hyperbolic parabolide, are defined by the grey levels, f(*x*, *y*), of the four pixels in the original image closest to *P*. This bilinear interpolation is defined by equation (13).

$$f(x,y) = ax + by + cxy + d \tag{13}$$

In our case, f(*x*, *y*) represents the interpolated grey level at position (*x*, *y*) in the original image. Four solutions to equation (13) are known, thus the four parameters (*a*, *b*, *c* and *d*) can be solved. The solutions of the equations are given by the positions and grey level of four pixels in the original image. The parameters are found by assuming that x=0 and y=0 at the point (x_c, y_c). This yields:

$$
\begin{aligned}
&f(x_c, y_c) = d \\
&f(x_c + 1, y_c) = a + d \\
&f(x_c, y_c + 1) = b + d \\
&f(x_c + 1, y_c + 1) = a + b + c + d
\end{aligned}
\tag{14}
$$

If the parameters are known then the grey level of a pixel in the transformed image can be calculated.
It is hard to judge how good this interpolation is from these equations. Another interpretation of bilinear interpolation, which leads to exactly the same results, makes it easier to show how accurate the method is, depending on the conditions.

Bilinear interpolation can be seen as an approximation of the area in the original image that defines the grey level of a pixel in the transformed image. The approximated area has the same size as a pixel in the original image. Figure 33 shows two times four pixels of the original image, with the area that defines the grey level of a transformed pixel indicated.

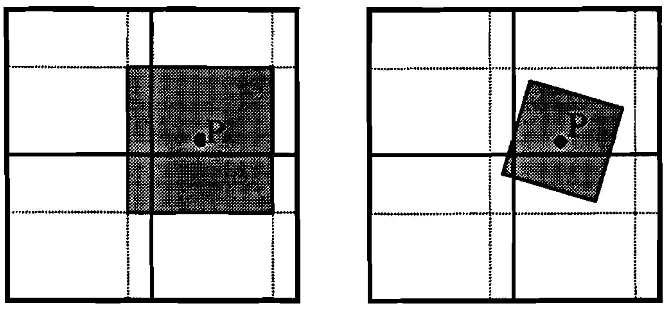


Figure 33. Bilinear interpolation.
Left: The approximated area. Right: The real area.

The area that defines the grey level of a transformed pixel is shown in the right image in figure 33. This area is approximated with the area shown in the left image in figure 33.
The overlap between the approximated area and the pixels in the original image is now easy to calculate.
How good the approximation is depends on the scaling factor and the rotation of the bilinear transformation. Up scaling causes the approximated area to be too large. This causes some blurring in the transformed image. Down scaling causes the area to be too small, which leads to loss of detail because not all relevant pixels are taken into account. In the extreme case this leads to nearest-neighbour interpolation. Rotations also cause loss of detail.
Here the bilinear interpolation is used to resample with scale factors close to 1 and small rotations, so the approximation should not be far off.

The information contained in a resampled image is always equal or less than in the original image. Sometimes it may seem as if a resampled image contains more details and thus more information. Figure 34 shows an example.

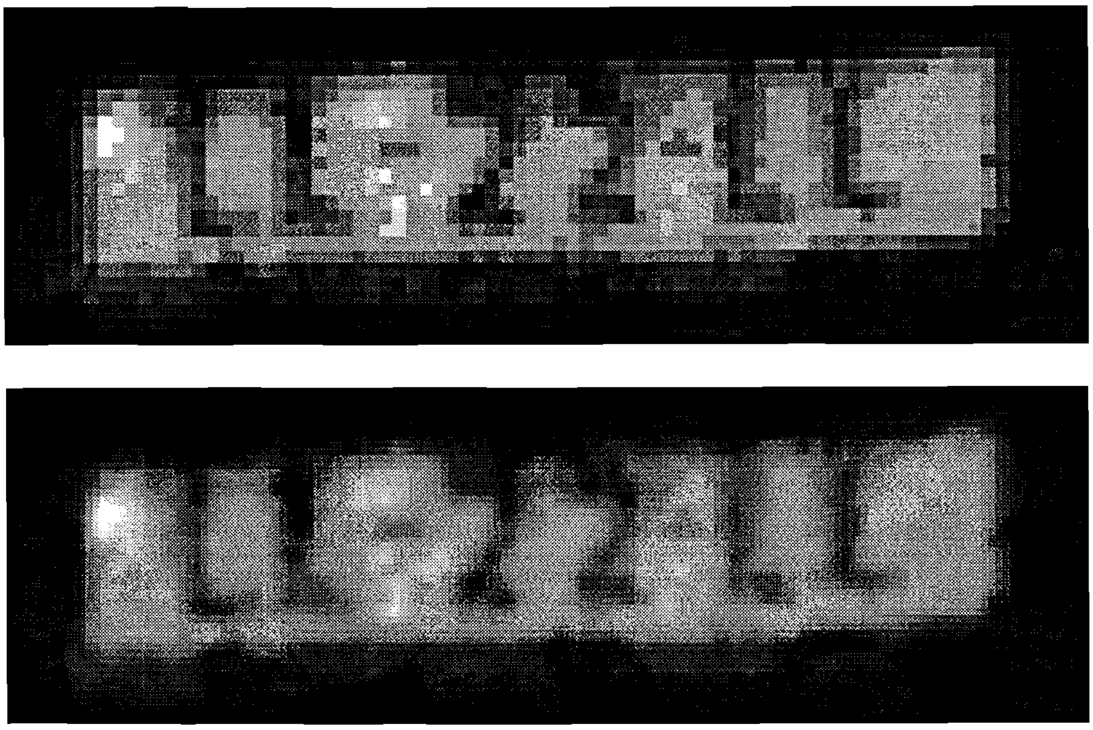


Figure 34. An original (top) and a bilinear resampled (bottom) image.

It seems as if the resampled image in figure 34 contains more details and thus contains more information, but this cannot be the case of course. Figure 35 shows another example of a resampled image.

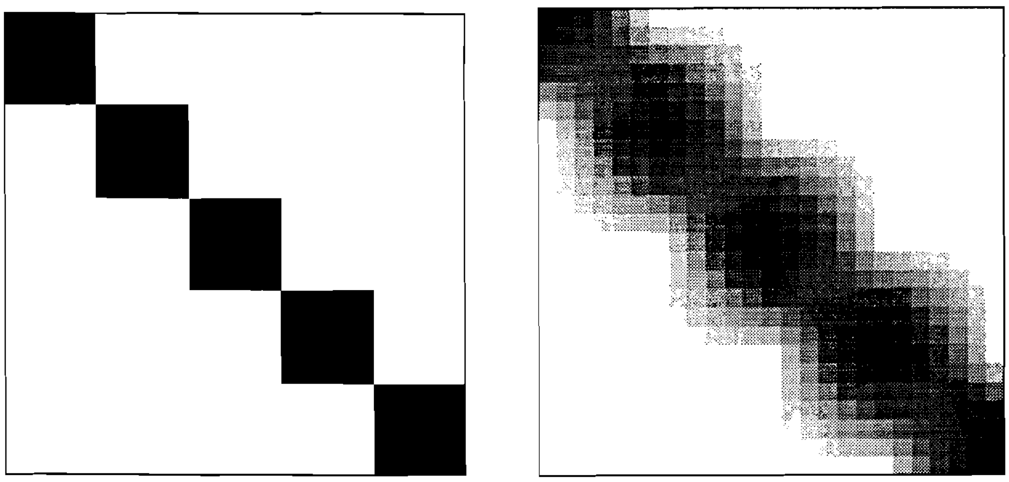


Figure 35. An original (left) and a bilinear resampled (right) image.

This example clearly shows the blurring caused by the bilinear resampling. The original image can not be reconstructed by resampling the blurred image.
A drawback of bilinear resampling is the introduction of blurring in the resampled image. Here this is not a real problem because the noise level in the license plate images is so high

that some blurring actually enhances the image. In this way salt and pepper noise is reduced somewhat.

6.3 Bilinear resampling

Bilinear resampling is used to scale the segmented license plate to fixed dimensions. From some sample images it was concluded that the dimensions of the plates are about 100 × 22 pixels. The dimensions of the destination image were chosen to be 180 × 40 pixels. This way all segmented plates are up scaled and the aspect ratio is about the same as in the source image. Up scaling also helps in finding the correct columns for the character segmentation. If the resolution of a plate is very low, as in some example images, then the gap between the characters is very small and hard to find. After up scaling this is usually no problem anymore.

The license plate segmentation had to be done by hand to get sample data. It was tried to achieve a segmentation accuracy according to the specifications given by CFT. Segmenting license plates can sometimes be very hard, especially if the image is very dark, like figure 28, or the area surrounding the plate is as bright as the license plate, as in figure 36. It is unknown how the system of CFT will react to these circumstances. It is assumed that either it will reject these plates or segment them according to the specifications.

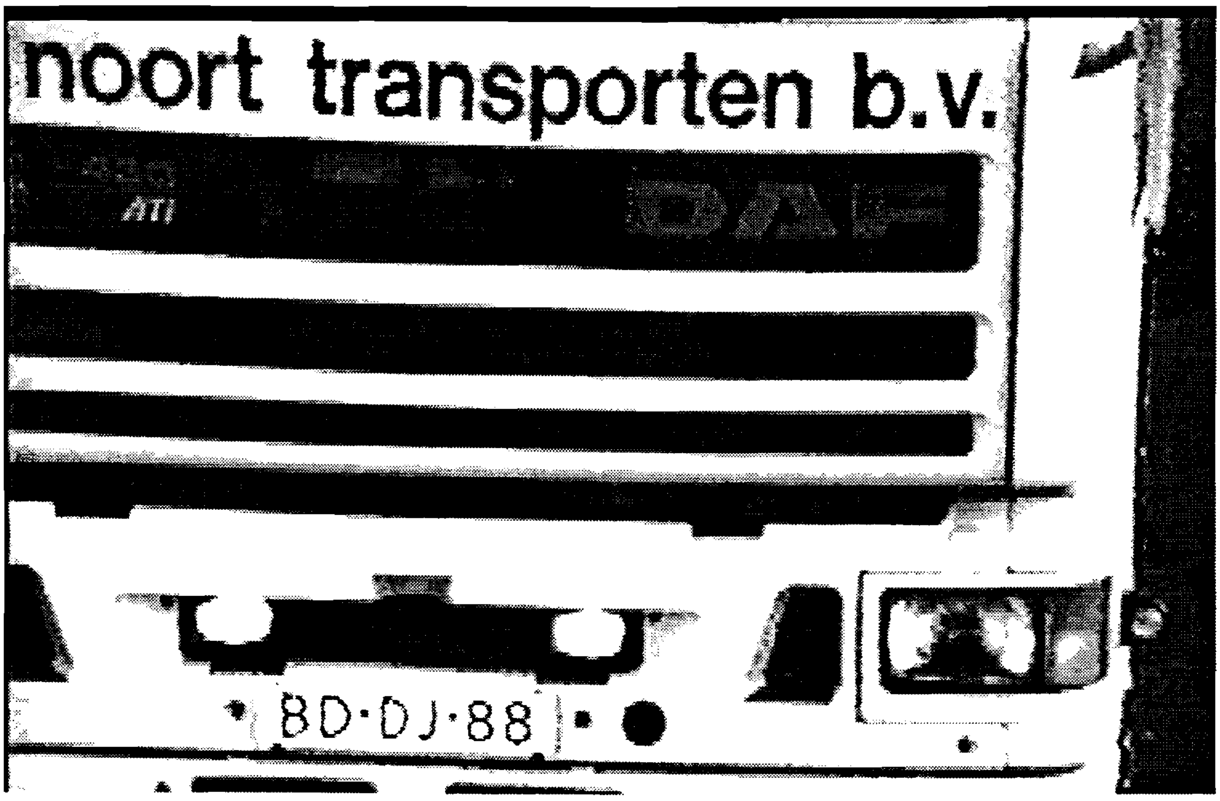
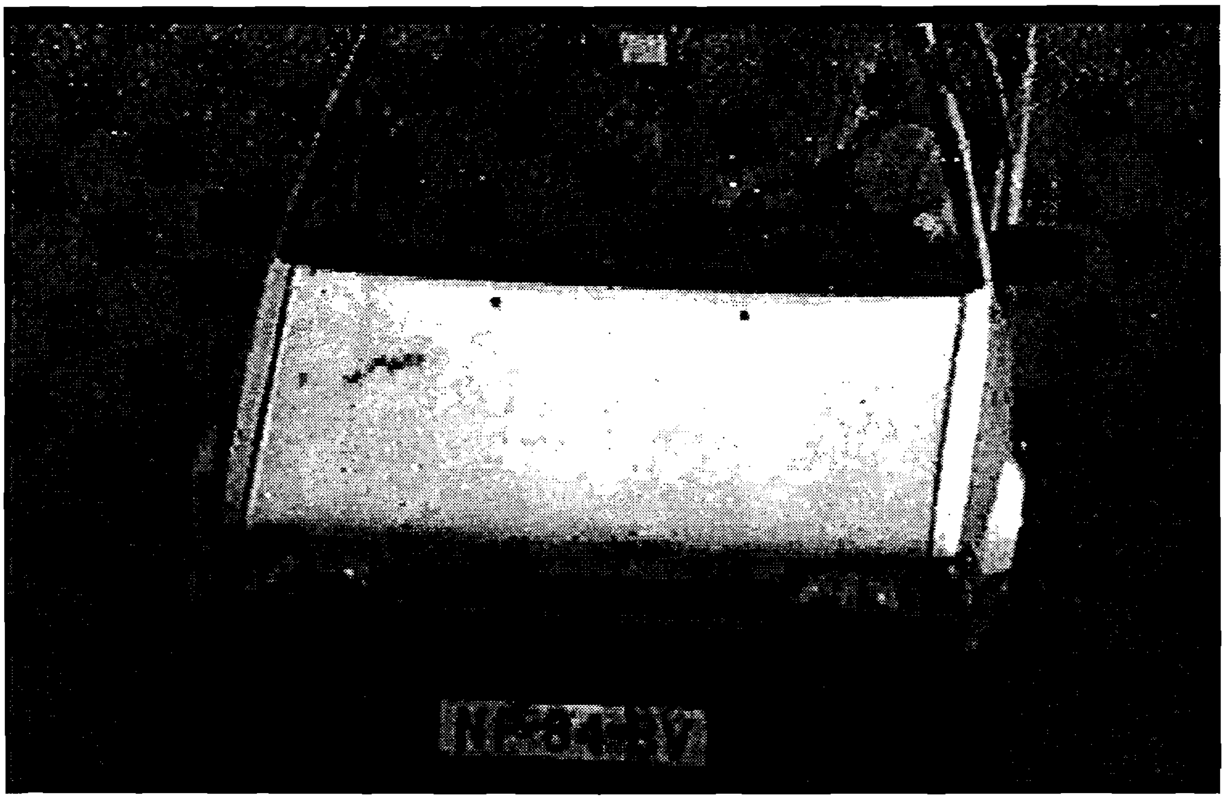


Figure 36. A difficult plate to segment.

An example of an original image and scaled and segmented license plate is shown in figure 37.



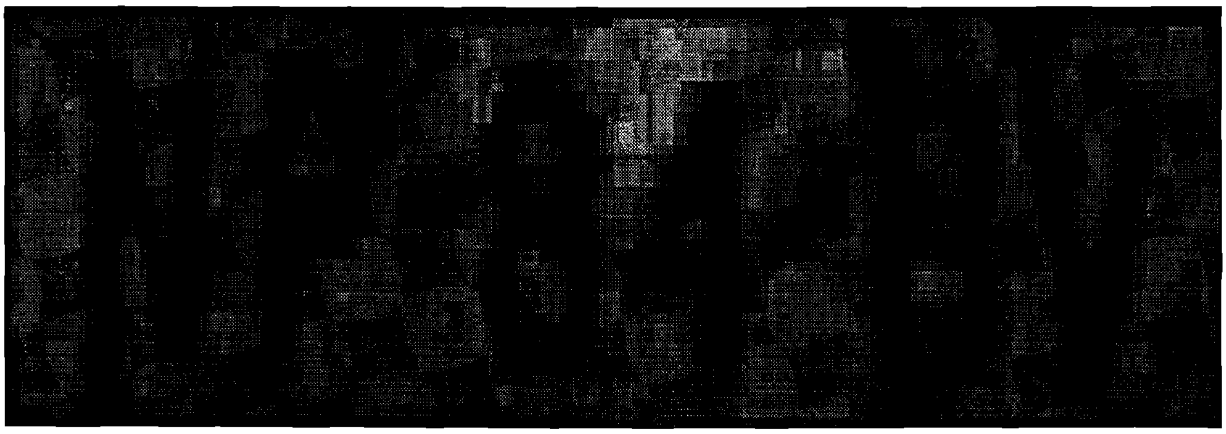


Figure 37. An original image and a segmented license plate scaled to 180 x 40 pixels.

Figure 38 shows how the bilinear transformation can be used to correct for perspective distortion.

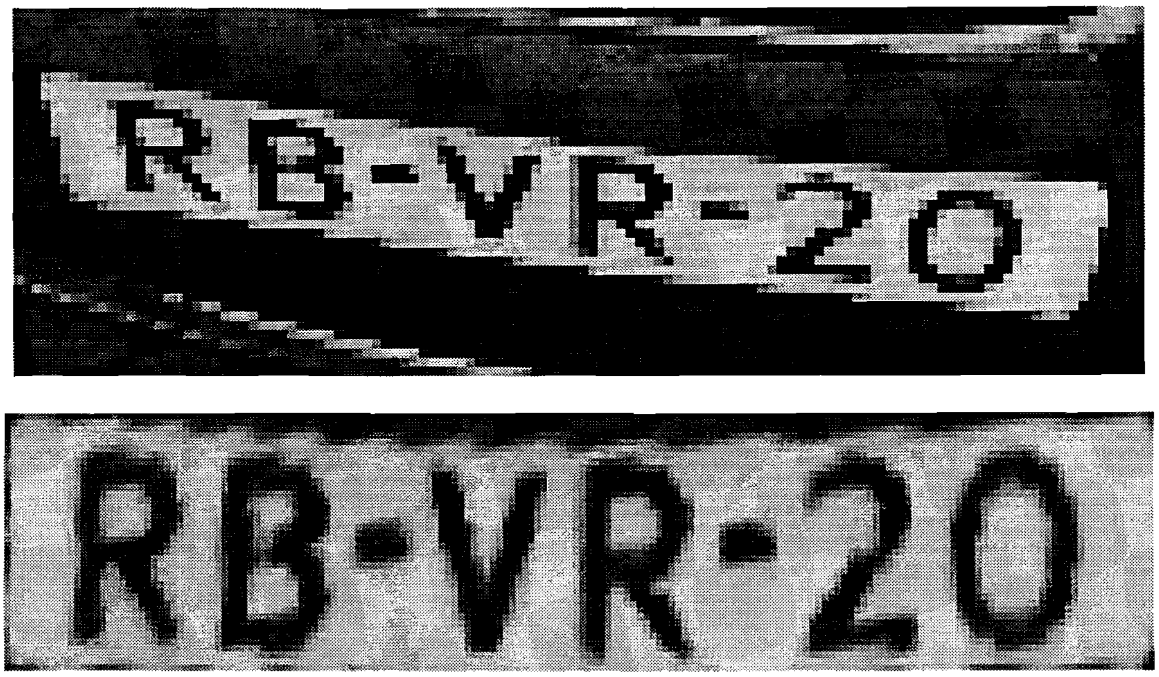


Figure 38. An image with perspective distortion (top) and the plate after the bilinear transformation and resampling (bottom).

This figure shows that the bilinear transformation can correct for most of the perspective distortion, but it cannot correct for the distortion that is caused by bent plates.

6.4 Conclusions and recommendations

At this time the license plate segmentation is done by hand. This leads to a lower rejection and error than with the automated system. Furthermore the segmentation accuracy is probably better when done by hand. To evaluate the system it would be best to have the automated system available.

The plates are extracted from the images with the bilinear transformation. This corrects for most of the perspective view distortion, but it cannot correct for distortions caused by bent plates. If the corner points of a plate are known then it should be possible to find the contours of the plate. Then a more advanced transformation, one that uses six points for example, can be used to find a better reconstruction of the plate.

The characters on a plate can be segmented after the license plate segmentation stage. This is done in the character segmentation stage, which is described in the next chapter.

7. Character segmentation

The role of character segmentation is very dependent on the classification method. There are classification schemes without segmentation, so-called segmentation-free classification schemes. Here another segmentation method for the common Dutch license plate is discussed. The contents of a Dutch license plate has to obey many rules. From these rules a model of a license plate can be made. This model just contains seven parameters, which are defined as ranges. Figure 39 shows this model.

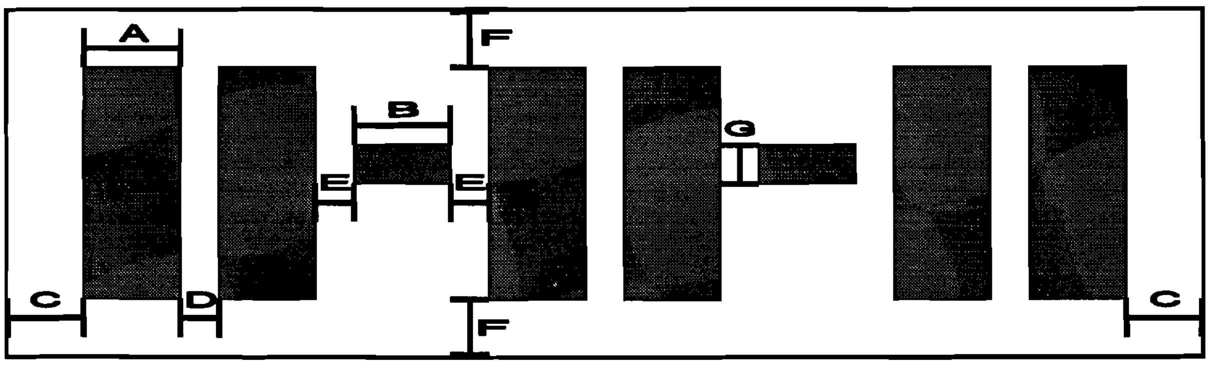


Figure 39. A model of the common Dutch license plate.

The vertical position of the dashes should be on top of the vertical middle of the plate. The next table gives the parameter description and its minimum and maximum value in pixels. The numbers given refer to the standard resolution of 180 × 40 pixels of a scaled license plate image.

Table 1. License plate parameters expressed in numbers of pixels.

Parameter	Description	Minimum	Maximum
A	Character width	8	20
B	Dash width	8	10
C	Side space	10	20
D	Character space	4	8
E	Character dash space	2	4
F	Top/bottom space	4	10
G	Character/dash line width	3	5

The ranges were initialised with values extracted from the official description of Dutch license plates. This description does not allow any freedom in some dimensions like for

example the line width. However, in practice it is easy to see that some dimensions are not the same for all plates. In these cases real live images were used to draw conclusions about the range of a dimension.
Many conclusions can be drawn from table 1. One can for example calculate the smallest possible distance between the left edge of the plate and the left side of the left dash. This is *not* calculated by summing the minimum possible values for *C A D A E*, because one dimension can influence another. Dimension *A* and *C* should not be equal to the minimum possible value at the same time. This is because the plate width is equal to $6A+2B+2C+3D+4E$, which is constant. If one wants to calculate the smallest possible distance between the left edge of the plate and the left dash then one has to imagine a plate that causes this distance to be minimal. This occurs if the characters in the left pair have the minimum possible character width and the other characters have the maximum character width.
The character segmentation stage is based on the column sum. The grey levels of all pixels in a column of a scaled plate are added up. By calculating the column sum it is tried to get information about the characters and the spaces between them, so only the areas where the characters actually can be should be taken into account when calculating this vertical projection. The minimum possible value for parameter *C* and *F* point out this area.
In figure 37 it is seen that a lot of noise is present, especially in the background of the plate. This is probably caused by dirt and reflections on the plate. A special form of histogram stretching can improve the image quality, resulting in a more reliable column sum graph. Histogram stretching is described in the next paragraph.

7.1 Histogram stretching

A histogram is a graph which shows the number of pixels of a colour in an image for each colour. Techniques like histogram equalisation, shifting, stretching and shrinking [VIII, p 6.1-6.2] can be used to enhance an image.
An ideal image should at least contain a certain percentage of light grey pixels from the yellow background of the plate. There should also be at least a certain percentage of dark pixels in the image. By examining some images it was concluded that an image should at least contain about 60% light pixels and about 10% dark pixels. Experiments showed that an image is enhanced for the character segmentation by projecting 60% of the lightest pixels to absolute white. Projecting 10% of the darkest pixels to absolute black proved to distort the character segmentation. If this is done then dark areas in between characters are sometimes projected to absolute black, thus degrading the column sum graph. Projecting just 3% proved to enhance the column sum graph. Linear histogram stretching is applied to the remaining 37% of the pixels. The described process of histogram stretching is illustrated in figure 40.

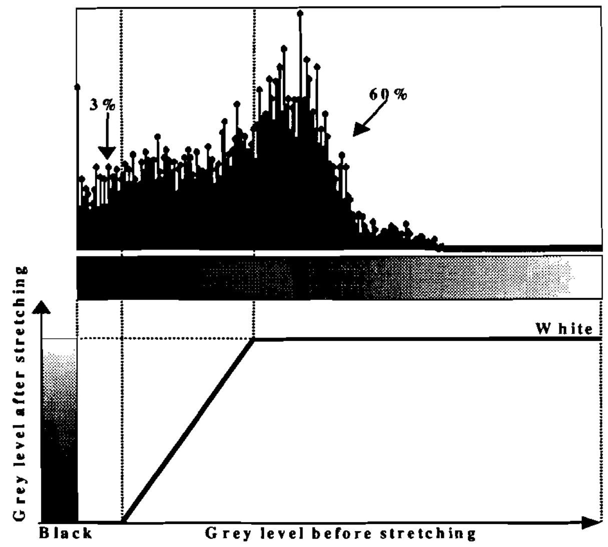


Figure 40. Histogram stretching for the character segmentation.

A histogram before and after histogram stretching is shown in figure 41.

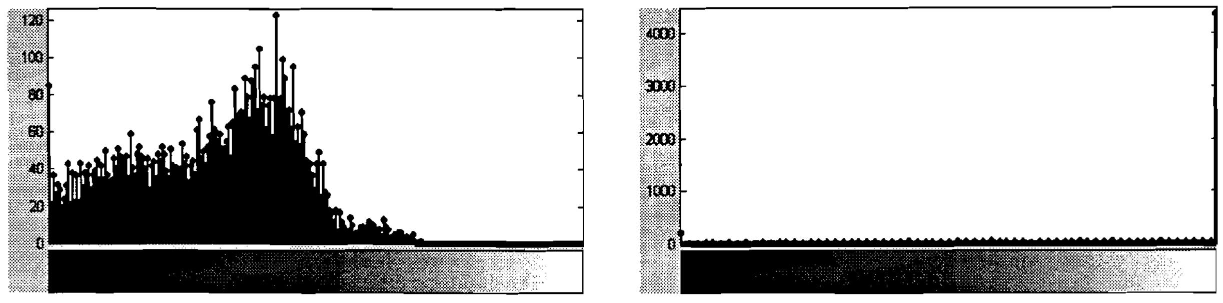


Figure 41. Left: The histogram of a license plate.   Right: The histogram after enhancement.

The enhanced histogram seems to indicate that the image was almost binarised, but keep in mind that the vertical scaling of the two images is different. Binarisation can actually occur with plates that have a high contrast. The most important part of the image is the dark part. This part is almost untouched by the enhancement. The major part of the noise in the background is removed by the histogram stretching. Figure 42 shows the plate of figure 37 after enhancement.

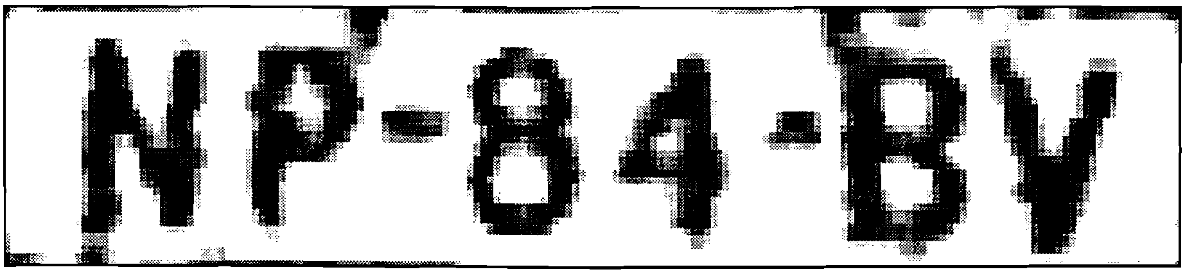


Figure 42. A for segmentation enhanced license plate.

This plate looks better than the original non-enhanced image, but that is not what is important here. The improvement that is achieved for the column sum graph is. Figure 43 shows the column sum graph of the license plate in figure 37, which is not enhanced.

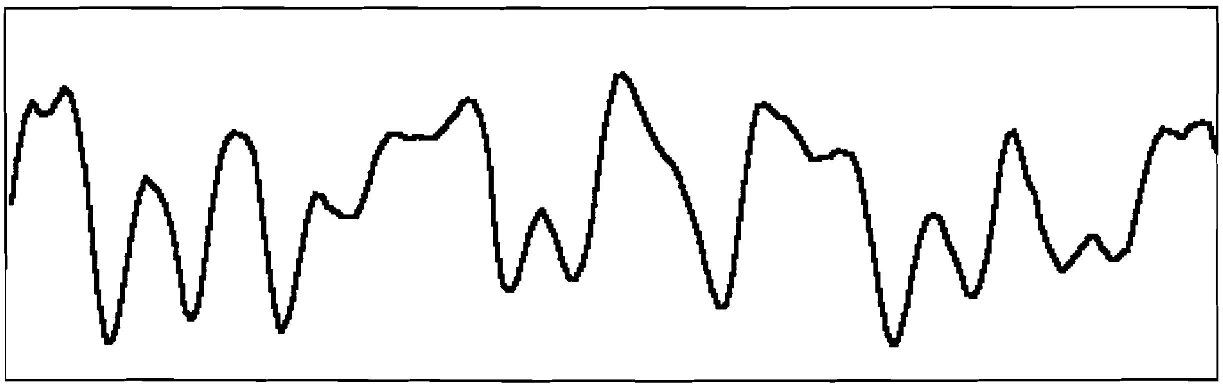


Figure 43. The column sum of a non-enhanced license plate.

The graph is very smooth which is very welcome here. The smoothness is caused by the blurring introduced by the bilinear interpolation.
Figure 44 shows the column sum graph of the same image after enhancement. The extremes in this figure are clearly more distinct.

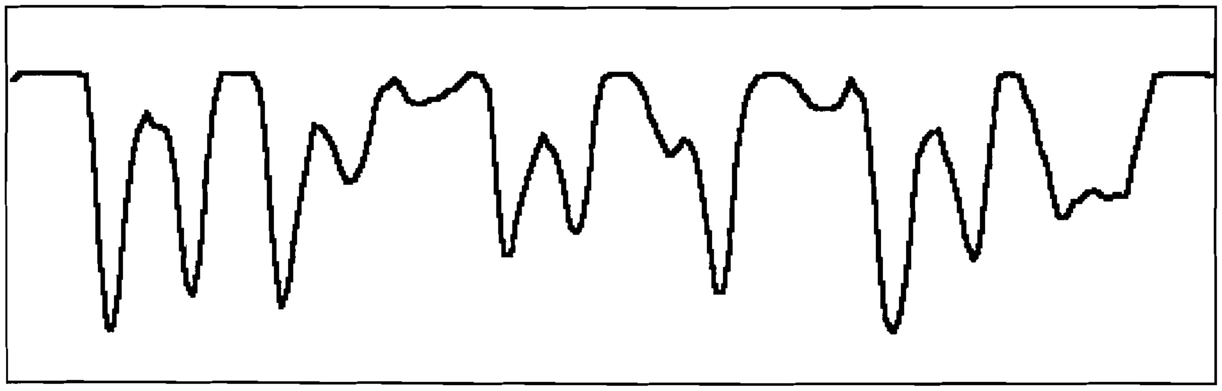


Figure 44. The column sum of the enhanced plate in figure 42.

This graph is analysed for extremes. This is done in the next step of the character segmentation.

7.2 Analysing the column sum graph

A graph like shown in figure 44 is examined for extremes. Dark pixels correspond with low grey levels so the characters and the dashes should produce minima in the graph. Spaces between characters and dashes should appear in the graph as maxima. The characters can also cause extremes in the graph.

A maximum is assumed if one, or both, of the following criteria are met.

- A value in the column sum graph is larger or equal to its six closest neighbours and the values four columns further and before the position are smaller than the basis position. This can be indicated with: "$< \leq \leq \leq = \geq \geq \geq >$", where each symbol represents a value in the graph. The "=" represents the maximum.
- A value in the graph between two minima is larger than a threshold, which is equal to 98% of the maximum possible value for the column sum. If more values next to each other exceed this threshold then only the centre position is assumed to be a maximum. If the number of positions is even, then there is no centre position. In this case the position to the right of the centre is assumed to be the maximum.

The second criterion is needed if the graph clips to the maximum possible value, like seen in figure 44. This happens if a large area of the background is projected to white. The definition of a minimum is analogous to the definition of the maximum. Figure 45 shows the extremes indicated in the column sum graph of figure 43.

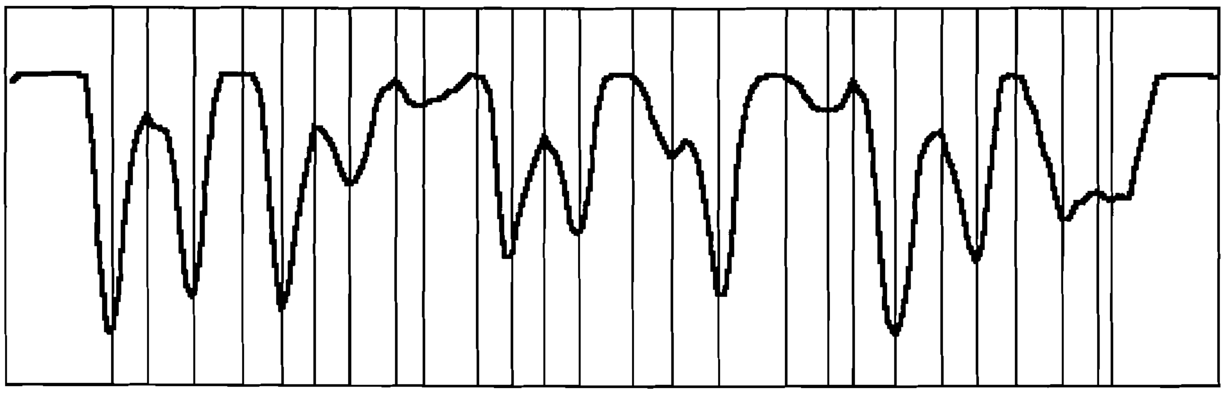


Figure 45. The extremes found in the column sum graph.

The definitions for the extremes are the results of fine-tuning with many plates. It is important not to miss extremes, on the other hand not every little distortion should lead to extremes. This could confuse the next steps.

7.3 Finding the dashes

Experiments showed that the step described here functions almost flawlessly for all plates tested. This makes this step the ideal starting point for the character segmentation.
From the model of the license plate it is concluded that the horizontal location of a dash is limited to a range. It is also known that a dash produces a minimum in the column sum graph. Template matching is applied to find the dashes. This is only done at the minima in the range for each dash. The best match in a range is assumed to be the location of the dash. Figure 46 shows a plate with its column sum and template results at the ranges.

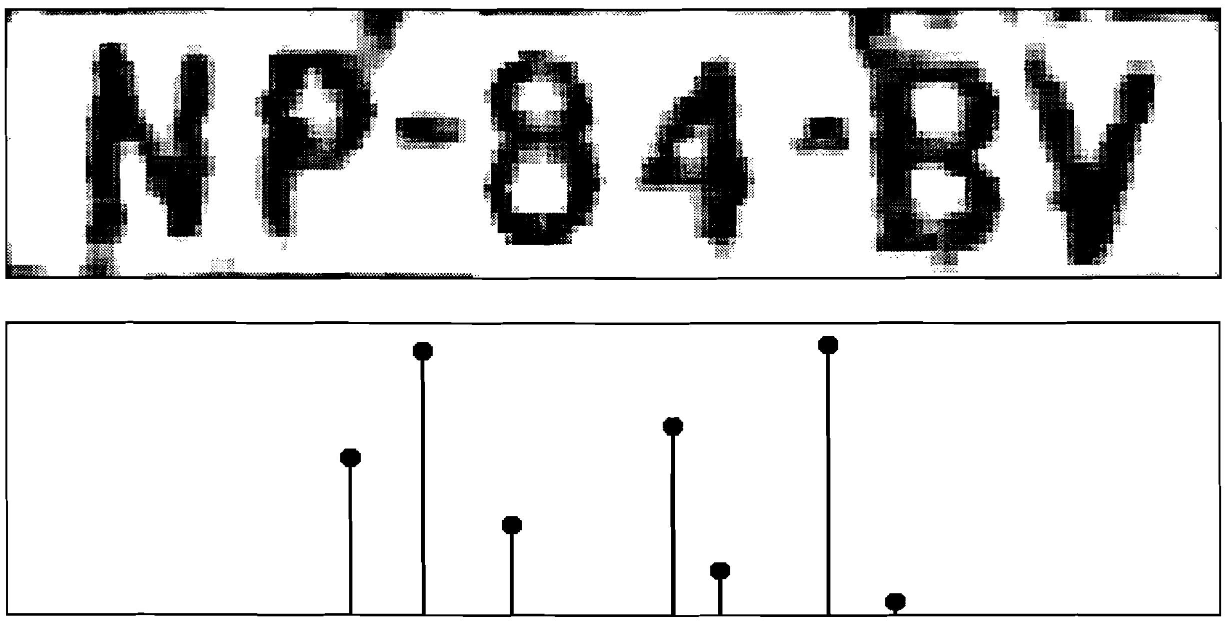


Figure 46. A plate (top) and the results of template matching at the minima in the dash ranges.

It is assumed now that a column containing part of a dash is indicated. Usually the dash is indicated somewhere in the middle. Next the dashes are segmented. This is done by comparing the indicated column with the columns in the neighbourhood. The columns that match best are assumed to be part of the dash. This way the dash is accurately segmented. Then the characters are segmented at the sides next to the dashes. This is done by shifting the segmentation positions for the dashes towards the characters by two columns. The number of columns to shift should actually be equal to the average number of columns between a character and a dash. Two columns proved to work nicely. At this moment a side of each character is segmented with an accuracy of about two columns. Figure 47 illustrates the procedure described.

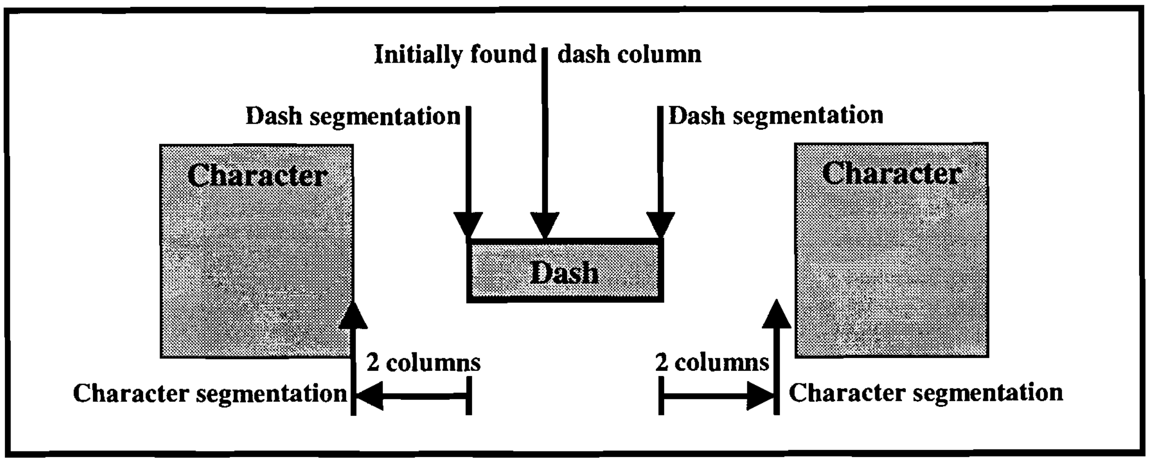


Figure 47. Estimating the character segmentation from the dash segmentation.

These steps work flawlessly for all examples tested but one can imagine some possible problems beforehand. A problem can occur if the dash does not produce a minimum or if part of a character is mistaken for the dash. Both cases are quite unlikely. A part of a character that can be mistaken for a dash, like for example the middle part of a “H” usually does not produce a minimum in the graph so it is ignored completely.

7.4 Segmenting the first and last character

The left side of the first and the right side of the last character are segmented by examining the column sum graph. Changes in the graph indicate the start or the end of a character. Let us assume that the left side of the first character must be found. The left of the plate contains an area that should be blank. The most right column in this area is the starting point. From there the changes in the column sum are calculated cumulatively. If the changes exceed a threshold then the start of a character is assumed. The right side of the last character can be segmented in the same way, then one start at the right and looks for changes to the left.
This method works quite well in most cases, but there are some problems. Figure 48 presents two cases that can cause some difficulties.

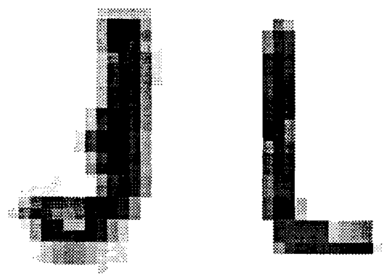


Figure 48. Two difficult to segment characters.

The changes in the graph at the columns where a character starts or ends are very dependent on the shape of the character. The left side of the "J" and the right side of the "L" produce only a small change in the graph. This change is very hard to detect, especially if there is noise in the background. This is partially solved by examining how steep the graph is in the area where the graph exceeds the threshold that indicates the start of a character. A steep graph indicates that a nice edge has been found. A flat graph indicates that probably a part of a character is cut off at the point found. The found position is corrected depending on how steep the graph is.
At this point the outsides of all character pairs are segmented. The next step is to segment them completely.

7.5 Segmenting the character pairs

The character pairs are segmented by examining the maxima in the column sum graph. A maximum is expected between the characters. The fact that all characters, except the "1" have about the same character width is used here. It indicates that a maximum in the middle of the two previously found character segmentation positions is most likely the right separator for the characters. The absolute level of the maximum is also taken into account. It is multiplied with the graph in figure 49, which gives a modelled estimation of the probability, in relation to the position, that a column is the right separator.

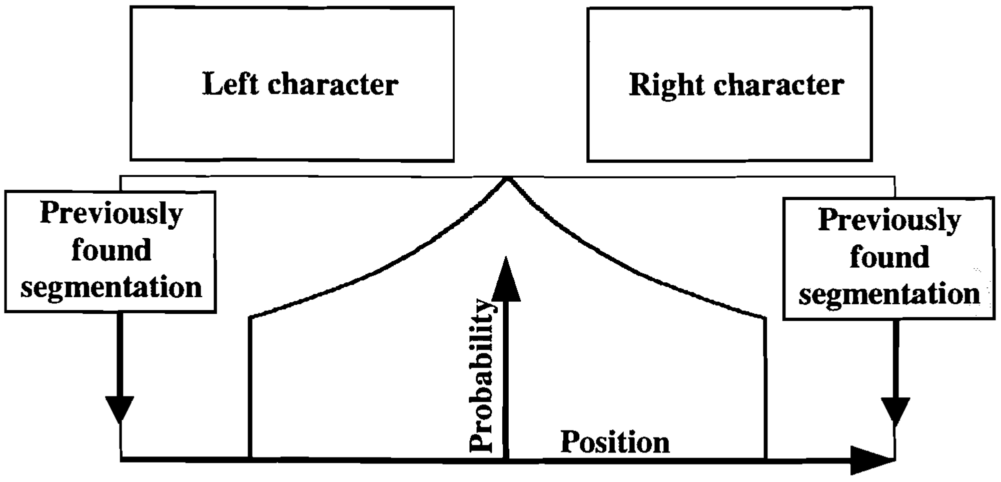


Figure 49. A modelled estimation of the probability of the segmentation position between a character pair.

It was not examined if the graph in figure 49 represents the actual probability, but it proved to enhance the character segmentation a great deal. It can be examined later if the segmentation needs to be enhanced further.

The probability is zero at the sides because of the minimum character width. In the middle it has a maximum. The found maxima in the column sum graph between the previously found character segmentations are weighted with the help of this graph. The highest result is assumed to be the separator between the characters. The space between the characters of a pair is five pixels on average. That is why the four neighbouring columns that match the found segmentation best are added to complete the horizontal segmentation. Figure 50 shows a plate with horizontally segmented characters.

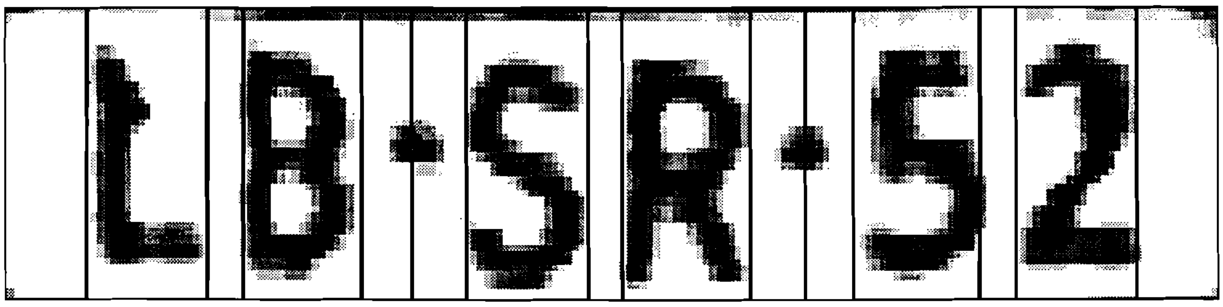


Figure 50. A plate with horizontally segmented characters.

The example shows an almost perfect segmentation. The next step is to segment the characters vertically.

7.6 Segmenting the top and bottom of the characters.

The vertical position of the characters on a plate should always be the same. However, because of the limited accuracy in the license plate segmentation and bent plates this position can vary. This can even cause different vertical positions for characters next to each other. Figure 51 shows an example of a bent plate.

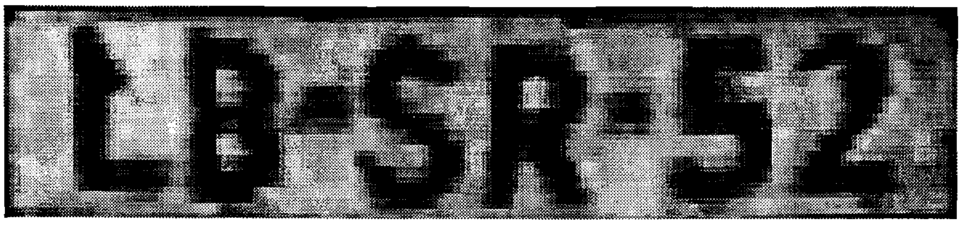


Figure 51. A bent plate.

It is clearly visible that the “L” in figure 51 has a different vertical position than the “B” next to it. A segmentation scheme is needed to find the vertical position of the characters. This is done with the help of an edge map. This way it is tried to find the outline of the characters. The threshold used for the edge map is tuned on the image. This is needed because of the

different circumstances, like sharpness and noise level, of the images. Figure 52 shows the edge map of the plate in figure 51. The threshold is tuned to find about 720 edge points. With this number most of the character outline is found.
The top and bottom of the characters produce horizontal lines in the edge map, as is seen in figure 52. The figure also shows a graph for the edge points in each row. The plate was enhanced before the edge map was derived.

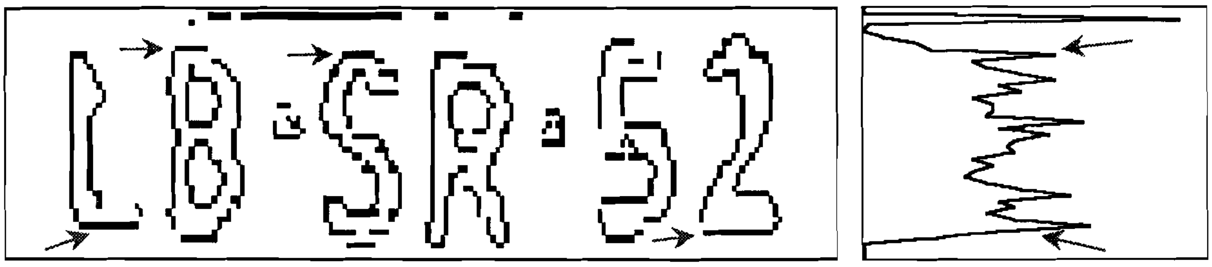


Figure 52. An edge map and its row sum for the edge point.

The start and stop row for the characters is estimated for the whole plate by looking at the cumulative row sum of the edge map. The right image in figure 52 points out the rows in the row sum graph that cause a high increase in the cumulative row sum. The row that causes this cumulative graph to exceed a threshold is assumed to be the start row. There is only a limited range of rows where a character can start or end. If the columns found are outside this range then the threshold is adapted until a possible solution if found. This approach differs very much from the strategy applied for the horizontal segmentation. Using edge information for the horizontal segmentation is very tricky, like shown in figure 53.

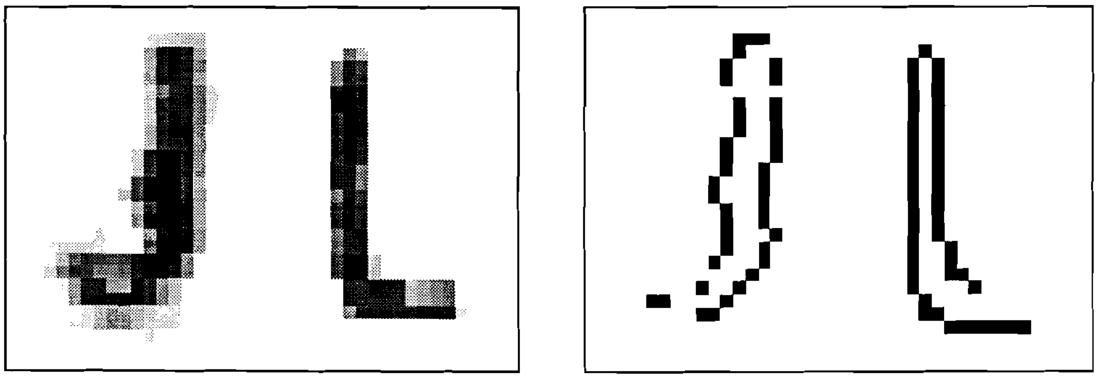


Figure 53. Two characters and their edge map.

Using edge information for the vertical segmentation relies on the edges of six characters, which is much more reliable than examining the edges of just one character. Smooth transitions from dark to light will not always be detected by an edge operation.
The found vertical segmentation is now fine-tuned on the character pairs. It is assumed that the vertical segmentation is maximally two pixels off. So five (two up and two down and the

estimated row itself) rows are taken into account to correct for this segmentation error. The rows are limited horizontally by the horizontal segmentation found earlier. The sum of these rows is calculated. The absolute level of the sum does not give much information. However, a change in the sum compared with a row up or down indicates the end of a character. The row with the highest change is taken as the correct vertical segmentation for the character. This is done for all characters, and functions very well in most cases. Figure 54 shows an example of a plate with fully segmented characters.

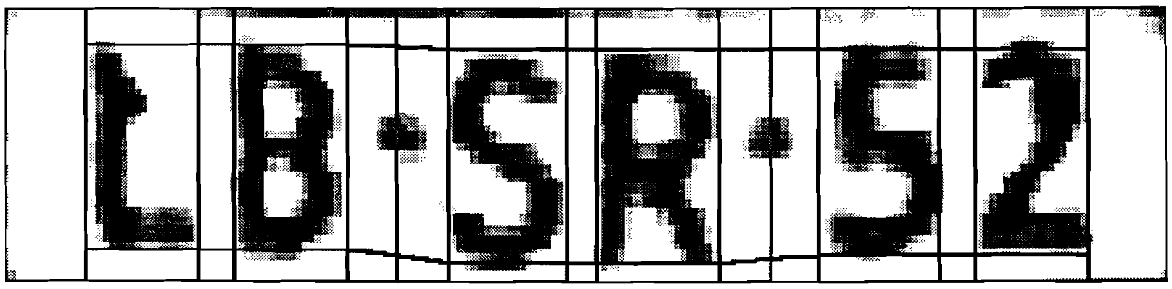


Figure 54. A plate with segmented characters.

The character segmentation was tested with 545 plates. Only 6 of them were segmented badly. The next figure shows an example of a segmentation fault.

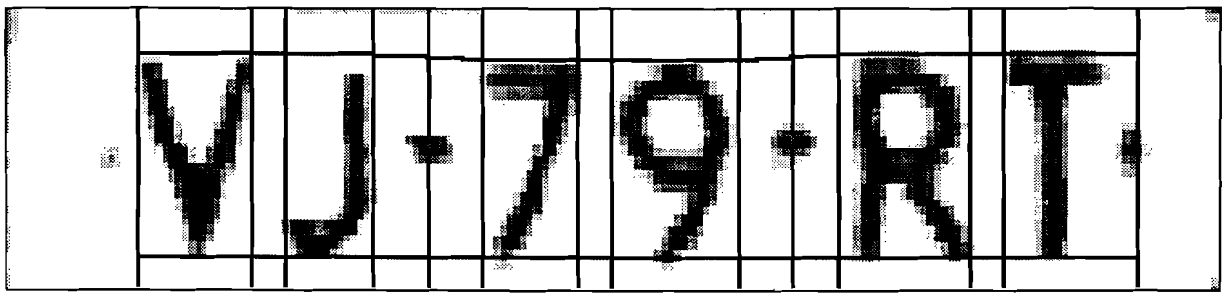


Figure 55. A segmentation error caused by a bolt.

The segmentation error in the last character in the plate of figure 55 is caused by a bolt to the right in the plate. It is hard to detect if this is part of a character of or not. The position of the bolts is usually in the vertical middle of the plate. Perhaps this property can be used to correct for this. The segmentation error in figure 56 is caused by the “1”. The segmentation between the characters of a pair is assumed to be most likely in the middle of the segmentations on the outside of the characters. Perhaps the model shown in figure 49 can be refined.

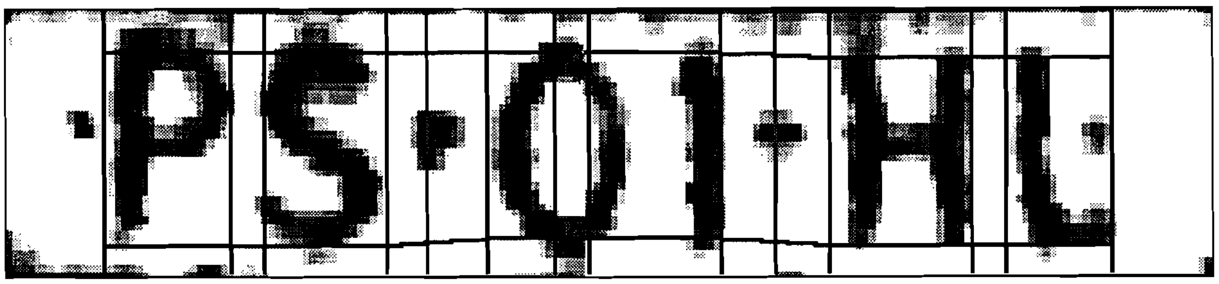


Figure 56. A segmentation error caused by the "1".

7.7 Conclusions and recommendations

The segmentation scheme described in the previous paragraphs performs very well and reliable in most cases. Only 6 of the 545 plates are segmented badly. The rest is segmented with an accuracy of about two pixels horizontally and one pixel vertically. This is acceptable for the classification, which is able to correct for these small errors.
A more accurate segmentation can probably decrease the error and rejection rate of the classifier. Here are some conclusions and ideas about how to improve the segmentation stage.

Some small changes in the parameters in the system can probably lead to a better performance. Especially the model of the license plate and the probability model shown in figure 49 can be refined.

Distortions like dirt and bolts can result in dark areas near the characters. This is the main cause of bad segmentation. Fine-tuning the segmentation parameters will probably only solve this problem partially. Also a more advanced image enhancement will probably not increase the performance much.

The plates are extracted from the images with the bilinear transformation. This corrects for most of the perspective view distortion, but it cannot correct for distortions caused by bent plates. If the corner points of a plate are known then it should be possible to find the contours of the plate. Then a more advanced transformation, that uses six points for example, can be used to find a better reconstruction of the plate.

Three properties that perhaps can help in the character segmentation are not used at the time. One is that the space to the left of the first character should be about the same as the space to the right of the last character. Also the property that the character height should be about the same for all characters can be used. The vertical segmentation of the character pairs should be about the same. Taking the average of the vertical segmentations of the characters in a pair is probably more reliable than examining them separately. It is better to have a 1-pixel error for both characters than to have a 2-pixel error for one and no error for the other.

A low misclassification rate is very important for the whole system. Bad segmentation can lead to misclassification. To reduce bad segmentation one can try to detect segmentation errors. The segmentation scheme actually does this but instead of rejecting the plate it tries to segment the plate again with adapted parameters. In most cases this corrects the initial error. Perhaps the classification stage can use this information.

At this moment the segmentation is done independently of the classification. This is not the optimum way to segment a license plate, especially if the classification stage is able to correct for small segmentation errors. A co-operation of the classification and segmentation stage is most likely to perform very well. One can for example imagine a scheme where a segmented character is classified. The classifier then enhances the segmentation or even rejects it. This information is fed back to the segmentation stage, which then responds accordingly.

The segmented characters are offered to the classifier. Here the classification is done with the help of the principal component analysis which is described in the next chapter.

8. The theory of the PCA transformation

The principal component analysis or Hotelling[6] transform is based on statistical properties of vector representation. It has several useful properties that make it an important tool for image processing. Any kind of information can be transformed. Here the transform is based on a set of prototype characters that each have to fit in a window of a fixed size. Each prototype is stored as a vector, the elements represent the grey levels of the pixels. The calculation of the transform itself is very simple, it just takes a few matrix manipulations. Some calculations on the prototype vectors have to be computed once before the transform can be computed.
The theory is the Hotelling transform is described in [III, p. 148-157], here this theory will be addressed again, but now the focus is on character recognition.

Consider a population of M prototype vectors x forming the $n \times M$ matrix X, with n the number of elements of a prototype vector. The elements of X represent the pixel data of the prototypes. The mean vector of the population is defined as

$$m_x = E\{x\} \tag{15}$$

The covariance matrix of the vector population is defined as

$$C_x = E\{(x-m_x)(x-m_x)^T\} \tag{16}$$

Where T indicates vector transposition. Because x is n dimensional, C_x and $(x-m_x)(x-m_x)^T$ are matrices of order $n \times n$. Element c_{ii} of C_x is the variance of x_i, the ith element of the x vectors in the population. Element c_{ij} of C_x is the covariance between the elements x_i and x_j of the x vectors. This yields $c_{ij} = c_{ji}$ thus C_x is symmetric and C_x is of course real because the pixel data is real. If elements x_i and x_j are uncorrelated then their covariance is zero, therefore in this case $c_{ij}=c_{ji}=0$.
For M prototype vectors, with n elements, stored as columns in the prototype matrix X, the mean vector and covariance matrix can be calculated with

$$m_{i,x} = \frac{1}{M}\sum_{k=1}^{M} x_{ik} \qquad i \in \{1,2,\dots,\mathrm{n}\} \tag{17}$$

and

$$C_x = \mathrm{cov}(X) = \frac{XX^T}{M} - m_x m_x^T \tag{18}$$

[6] This transform is commonly referred to as the principal component analysis, eigenvector or discrete Karhunen-Loève transform.

C_x is always real and symmetric, thus[7] C_x has an orthonormal set of n eigenvectors.
All λ's that satisfy the next equation are the eigenvalues of matrix A.

$$Ae = \lambda e \tag{19}$$

The corresponding e's are the eigenvectors of A. It is obvious that any λ will satisfy equation (19) if e is the zero vector, these λ's are not allowed, but λ =0 is a valid eigenvalue.

The M prototype vectors form a basis in R^K with $K \leq M$. Equation (18) yields that C_x also forms a basis in R^K. With respect to the eigenvalues this leads to the conclusion[8] that no more then M distinct eigenvalues can exist for C_x. Still the eigenvectors of C_x form a basis in R^n with n the number of elements in a prototype vector. Because C_x forms a basis in R^K and the eigenvectors form a basis in R^n, it is clear that at least n-M λ's are zero. Furthermore, if there are M vectors that point to positions in R^L, with $L \geq 1$, then it is possible to construct a basis of order M-1 in which it is possible to point out all M positions. This yields that at least n-M+1 eigenvalues are zero. For example, to point out two positions in 3-D, a basis of order one is needed. This is of course a line.
Now let e_i and λ_i with $i \in \{1, 2, ..., n\}$ be the eigenvectors and corresponding eigenvalues of C_x arranged so that $|\lambda_j| \geq |\lambda_{j+1}|$ with $j \in \{1, 2, ..., n\text{-}1\}$. Equation (19) yields that

$$C_x E = E\mathrm{H} \tag{20}$$

with H the diagonal matrix of the ordered eigenvalues and E a matrix containing the first n-1 eigenvalues as columns. Let K=M-1, then at least n-K eigenvalues are zero, equation (20) yields

$$C_x E = E\begin{bmatrix} \lambda_1 & & & & & & 0 \\ & \lambda_2 & & & & & \\ & & \ddots & & & & \\ & & & \lambda_K & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ 0 & & \cdot & & & & 0 \end{bmatrix} \tag{21}$$

The prototype matrix X, forms a basis in R^K, with $K \leq M$-1, so C_xX , a $n \times n$ matrix, will be of the form:

[7] A symmetric $n \times n$ matrix has an orthonormal basis of eigenvectors in R^n. [Kr, p. 436]
[8] If an $n \times n$ matrix A has n distinct eigenvalues, then A has a basis of eigenvectors in R^n.*[Kr, p. 435]*

$$C_x X = \begin{bmatrix} y_{11} & \cdots & y_{1n} \\ \vdots & & \vdots \\ y_{K1} & \cdots & y_{Kn} \\ 0 & \cdots & 0 \\ \vdots & 0 & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \tag{22}$$

Usually n will be much larger than K, so a very large part of the matrix will always contain zeros hence these elements do not have to be calculated.

Now let A be a matrix whose rows are formed from the first K eigenvectors of C_x so that the first row of A is the eigenvector corresponding to spectral radius λ_1. Matrix A is then a $K \times n$ matrix. Suppose that A is a transformation matrix that maps the x's into vectors denoted by y's as follows:

$$y = A(x - m_x) \tag{23}$$

Equation (23) is called the principal component transform. The y's are K dimensional. The mean of the y vectors resulting from this transformation is zero because the mean of the ith elements of $(x - m_x)$ over the population is zero, thus,

$$m_y = 0 \tag{24}$$

The covariance matrix of the y's can be obtained in terms of A and C_x by

$$C_y = YY^T - m_y m_y^T = YY^T = AC_x A^T \tag{25}$$

Furthermore, C_y is a diagonal matrix because Y is an orthogonal matrix. The elements of C_y along the main diagonal are the eigenvalues of C_x, that is,

$$C_y = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix} \tag{26}$$

The off-diagonal elements of the covariance matrix are zero, so the elements of the y vectors are uncorrelated. Keep in mind that the λ_j's are the eigenvalues of C_x and that the elements along the main diagonal of a diagonal matrix are its eigenvalues. Thus C_x and C_y have the same eigenvalues. In fact, the same is true for the eigenvectors.
Figure 57 illustrates the concepts just discussed.

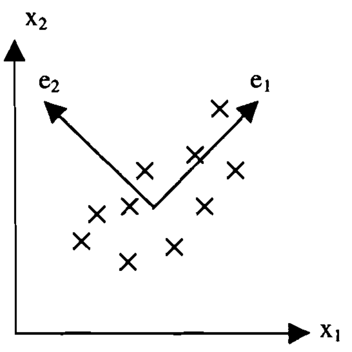


Figure 57. A 2-D population with its principal axes.

The binary object shown is treated as a 2-D population. In other words, each cross in the figure is treated as a 2-D vector $x=(a,b)^T$, where a and b are the coordinate values of the crosses with respect to the x_1 and x_2 axes. These vectors are used to compute the mean vector and covariance matrix of the population. The net effect of using equation (23) is to establish a new coordinate system whose origin is at the centroid of the population and whose axes are in the direction of the eigenvectors of C_x, as shown in the figure. This coordinate system clearly shows that the transformation is a rotation transformation that aligns the data with the eigenvectors of C_x. In fact, this alignment is precisely the mechanism that decorrelates the data. Furthermore, as the eigenvalues appear along the main diagonal for C_y, λ_i is the variance of component y_i along eigenvector e_i. The concept of aligning an object with its principal eigenvectors plays an important role in image analysis.
An important property of the Hotelling transform deals with the reconstruction of x from y. The inverse of an orthogonal matrix is equal to the transposition of the matrix, so $A^{-1}=A^T$. A vector x can be reconstructed from its corresponding y by using the relation:

$$x = A^{-1}y + m_x = A^T y + m_x \tag{27}$$

If instead of using all the eigenvectors of C_x only the first K eigenvectors are used to form matrix A then the $K \times n$ transformation matrix projects the input vector onto the first K eigenvectors forming a basis in R^K. The reconstruction given in equation (27) will be exact for vectors that are in the prototype population. The same is true for all vectors being linear combinations of the prototype vectors. All other vectors cannot be reconstructed exactly.
It can be shown that the mean square error between a reconstruction that uses all n eigenvectors of C_x and one using only the first K eigenvectors can be expressed by:

$$d = \sum_{j=1}^{n}\lambda_j - \sum_{j=1}^{K}\lambda_j = \sum_{j=K+1}^{n}\lambda_j \tag{28}$$

As was shown at least n-K eigenvalues are zero, so the mean square error is zero for the reconstruction of a transformation of a prototype vector or of a linear combination of the prototype vectors.

In words the PCA transformation can be described like:

- A PCA transformation is an orthonormal decomposition by projection onto the relevant eigenvectors of the correlation matrix. Thus the eigenvectors point in the directions of the maximum energy/variation of the prototypes. This way only the information that distinguishes one prototype from another is kept. This leads to a reconstruction that is not exact anymore for characters that hold information that is irrelevant for classification.

The next chapter will examine how this theory can be used to classify characters.

9 Classification with the PCA transformation

The character segmentation offers a segmented plate to the classification stage. This plate is first enhanced with the histogram stretching described in paragraph 7.1. However, this time only 30% of the lightest pixels is projected to white. 3% of the darkest pixels is projected to black again. These values leave the grey areas surrounding the characters in tact. These areas also contain information about the characters. Then the character size, brightness and contrast are normalised. The character size is normalised based on the character height because this is most accurately found by the character segmentation stage. The character is resampled to a fixed height. The width is scaled with the same amount as to keep the same aspect ratio. Then the brightness and contrast of the resulting character is normalised. Of course the prototypes are treated in the same way. An average prototype for each class is calculated. Not all characters are used in Dutch license plates. Here some sample characters that are used very rarely are ignored because no or only one or two samples were available. A more detailed description about how this is done can be found in paragraph 10.2.1 .
With these prototypes the PCA transformation matrix is calculated. This transformation projects the prototypes to R^K, with K equal to the number of prototypes minus one. In this K-dimensional space only the relevant information about the prototypes is kept. All information that is not needed to distinguish between the prototypes is lost. The transformation will project all sample characters onto the prototypes hyper plane as is shown in figure 58 for the 2-dimensional case.

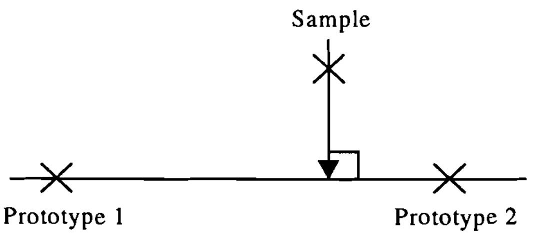


Figure 58. A sample character that is projected by the PCA transformation.

The figure shows an example with just two prototypes that are defined in two dimensions. The prototype positions correspond to the PCA transformation of the prototypes. All 2-D vectors, like the sample vector, will be projected orthogonal onto the line as shown in the figure.
If the sample vector is not on the line then the reconstruction will not be exact, this means that irrelevant information in the sample vector is filtered out. With irrelevant information all information is meant that does not actually cause a sample character look more or less like one of the prototypes.

On the line it is easy to measure how much the sample looks like the prototypes, only the Euclidean distance between the projected point and the PCA transformation of the prototypes has to be calculated. The distance is a valid property to measure how much a sample vector looks like a prototype because the eigenvectors of the covariance matrix form an orthonormal basis. If only the distances between the PCA transformations are needed then equation (23) can be simplified to

$$y = Ax \tag{29}$$

This will remove an offset from the PCA transformations.

A reconstruction of a PCA transformation can even look 'better' than the original sample character, artefacts like noise that aren't in the prototype plane are filtered out. However, for classification the reconstruction is not better then the original, only parts of the character that are irrelevant in terms of classification are filtered out. This will be illustrated in the next example. Figure 59 shows imaginary prototypes. The prototypes are assumed black and white for simplicity. The black squares represent values of 0, the white represent values of 1.

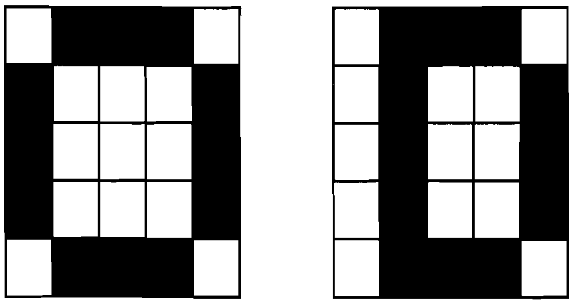


Figure 59. Two imaginary prototypes, a "O" and a "D".

Then the average prototype looks like:

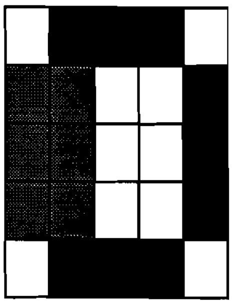


Figure 60. The average prototype.

The grey area represents values of 0.5.

Now let us assume that a sample character like figure 61 must be classified.

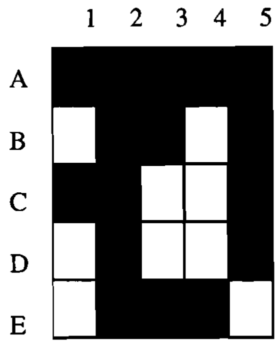


Figure 61. A sample character.

The parts that the two prototypes have in common are irrelevant because they do not help in distinguishing the two prototypes. The relevant squares are the squares that are different for the prototypes being *B1*, *B2*, *C1*, *C2*, *D1* and *D2*. In order to measure how much the sample character looks like one of the prototypes, just one measurement is needed because there are two prototypes. This measurement is represented by the first eigenvector of the covariance matrix, which is also called the first PCA component. Figure 62 shows the first eigenvector.

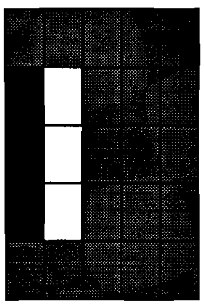


Figure 62. The first eigenvector.

The calculation of a PCA transformation can be interpreted as a special kind of "feature" extraction with the help of the eigenvectors of the covariance matrix of the prototypes.
The prototype "D" can be expressed as the average prototype minus the first PCA component. It can be calculated that the PCA transformation of the prototype "D" and "O" are -1.22 ($-\sqrt{6}/2$) and 1.22($\sqrt{6}/2$) respectively. The PCA transform of the sample character will be -0.82($\sqrt{6}/3$). Thus the (PCA)distance between the prototypes is about 2.4 and the distance of the PCA transform of the sample character and the "D" is just 0.4, yielding that the sample character is probably a "D". The reconstruction is given in figure 63.

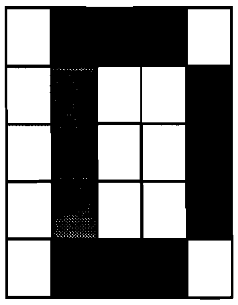


Figure 63. The reconstruction of the sample character.

The reconstruction looks better like the prototype "D" than the original sample vector in figure 61. However, just two things have happened, the black squares that are not part of one of the prototypes are removed. In addition, the parts that overlap the black and white area in the first component are spread out.
The approach described here is segmentation dependent. The next paragraph will examine the segmentation dependency closer.

9.1 Segmentation dependency of the PCA transformation

The classifier has to deal with small segmentation errors. So one could wonder what would happen if the sample character in figure 61 is segmented a little differently? Let's assume that this results in a sample character that looks like figure 64, which is the sample character of figure 61 shifted one pixel to the left.

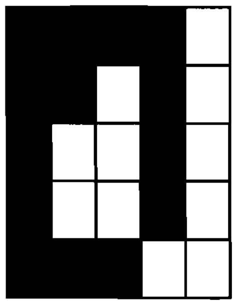


Figure 64. A sample character.

The PCA transformation of this character is $0.82(\sqrt{6}/3)$, which is closest to the PCA transformation of the prototype "O". So now the character is classified as an "O". This segmentation dependency is very disturbing for the classification, counter measures have to be taken.
This is discussed in the next paragraph.

9.2 Segmentation error correction

The segmentation offered by the character segmentation stage contains an errors of about two pixels horizontally and one pixel vertically. However, an accurate segmentation is needed to get the right classification result. An accurate segmentation can be achieved by shifting the character in the segmentation window over several positions. The character is shifted two positions to the left and right and one pixel up and down. Now one of these positions should contain an optimal segmentation, but which one is it? The next example will illustrate how this position is found. Figure 65 shows the reconstruction of the PCA transformation of sample character in figure 64.

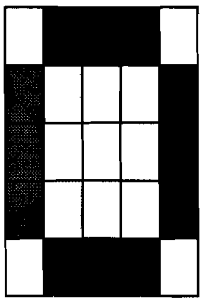


Figure 65. The reconstruction.

If the sample character of figure 64 is compared with its reconstruction in figure 65, then one can see that many changes have taken place. If the same is done with figure 61 and its reconstruction in figure 63, then one can see that there are fewer differences. These differences can be measured as a distance. A big distance indicates a possible segmentation error and thus a possible classification error because the PCA transformation then ignores much of the sample character. Especially large segmentation errors are detected by this distance measure. Small segmentation errors will not result in classification errors as quickly. Many distance measures are possible. It seems only logical that big differences in pixel value are weighted heavier than small ones. Small differences could even be ignored because they are probably caused by noise. Large differences indicate that a part of a character is missing or was added.

The distance measure calculates the "distance" between the individual pixels of the original character and the reconstruction of the transformed character. This could be done like:

$$D = \sum_{i=1}^{n} \left| x_i - r_i \right|^P \tag{30}$$

With x_i respectively r_i the grey levels of the input and the reconstructed character. The number of pixels is n. A good value for P was found with some experiments. The values 0.5, 1, 1.5, 2, 2.5 and 3 were tried. P equal to 2 was found to give the best results. With this value the distance measure delivers a good measurement to predict bad segmentation.

Differences in pixel value close to each other should contribute more to the distance measure than scattered ones. Scattered differences can be caused by noise like for example salt and pepper noise. This should not contribute much to the distance measure. Differences that occur close to each other indicate that part of a character is missing or is added. This can be detected by low-pass filtering the reconstruction subtracted from the original character. A Gaussian filter was tried with different spreads and kernel sizes. However none of the experiments improved the distance measure.
Resuming, to correct for a segmentation error the character is shifted up and down and to the sides. Two positions to each side and one position up and down proved to work best. Also trying a segmentation without the first and last row improved the system. This last segmentation is only shifted to the sides. So in total there are 20 positions tried. At first more positions to the side and up and down were tried, this only slightly improved the rejection ratio, but introduced some extra misclassification errors. All tried positions are PCA transformed and reconstructed. The original and the reconstruction are used to measure the distance. Then only the six segmentations with the smallest distance measure are taken into account as possible correctly segmented characters. The PCA transformation does not care for data that is the same for all prototypes, it ignores these areas. So it would be best to segment the character as close as possible. However, for the distance measure this is not ideal. Especially badly segmented characters will betray themselves if the window for the prototype is only a little larger than the prototypes. Only the parts that are in the window are taken into account to measure the distance between an original and a back transformed character. So the window for the prototypes is chosen a little larger than is needed for the largest prototype. The prototypes were derived from the average per class of all(2766) characters of the set of good license plates. An example of a prototype is shown in figure 66.

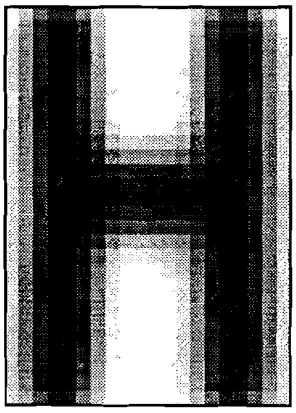


Figure 66. The prototype "H".

Figure 67 shows a character in different positions with its reconstructed version.

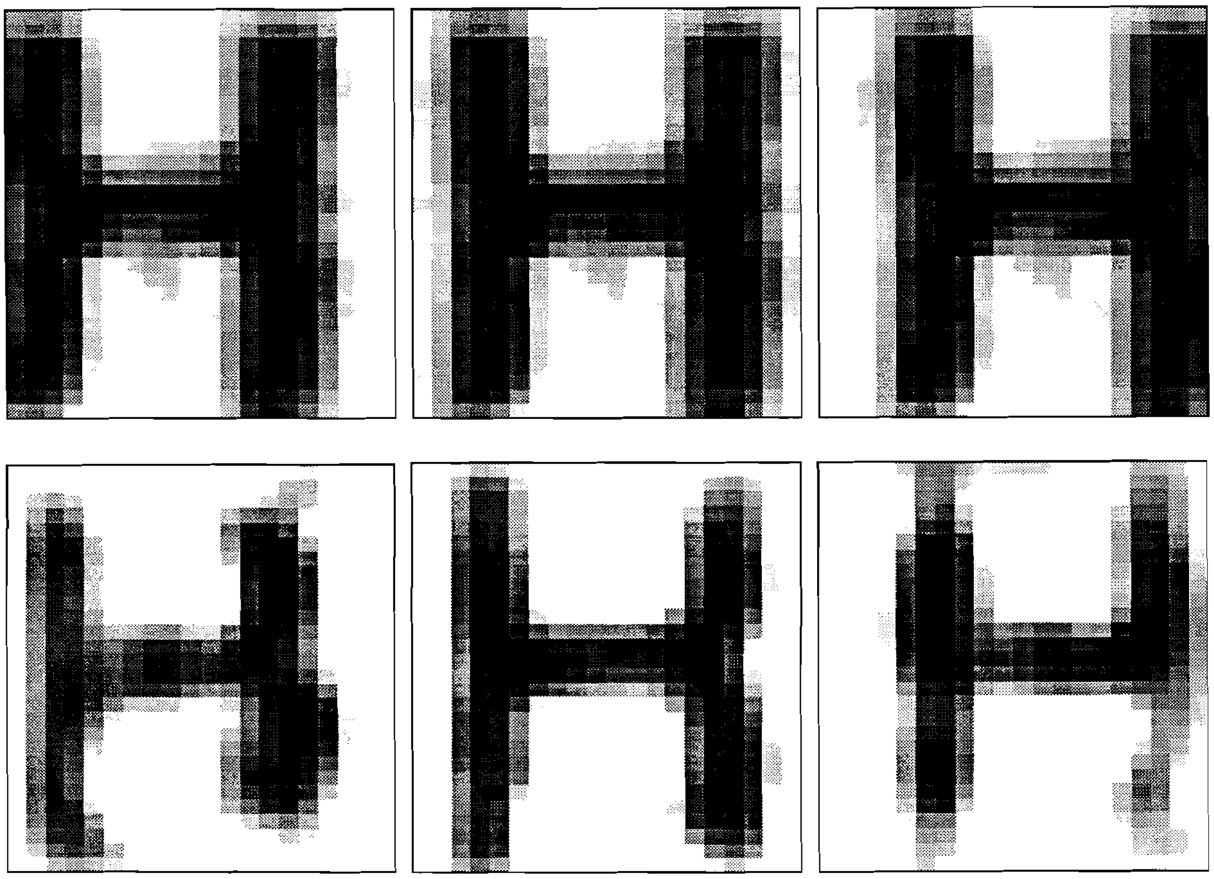


Figure 67. A sample character (top) in different positions with its reconstruction (bottom).

This figure shows that a difference between the original and its reconstruction can indicate a segmentation error. This way especially large segmentation errors are detected easily, but also other kinds of “distortions” can cause a differences. So this distance measure indicates an optimal segmentation and thus the PCA transformed character holds the right information for classification. How a PCA vector can be classified is examined in the next paragraph.

9.3 Classification of the a PCA transformed character

The PCA transformation of characters and the prototypes are points in n-dimensional space. Two important properties of this space are known.

- The Euclidean distance between two points gives a measure of the relevant distortion between the corresponding characters.
- The variation in the consecutive elements of a PCA vector tends to decrease.

From this one can conclude that the Euclidean distance can be used to classify PCA transformed characters and that not all elements of a PCA vector hold as much information. This is illustrated in figure 68 for the PCA transformation of the prototypes.

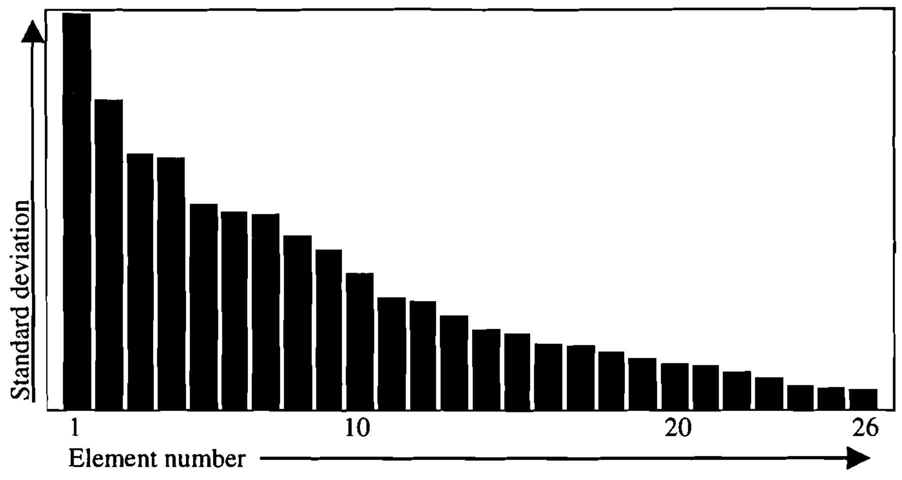


Figure 68. The standard deviation of the 26 consecutive elements of the PCA transformed prototypes.

The figure clearly shows that the standard deviation, and thus the ability of discrimination, decreases for the consecutive elements for the prototypes. So one could wonder how the reconstruction of a PCA transformed prototype would look if not all elements are used. Some examples of this are shown in figure 69 for the prototype "0" and "K".

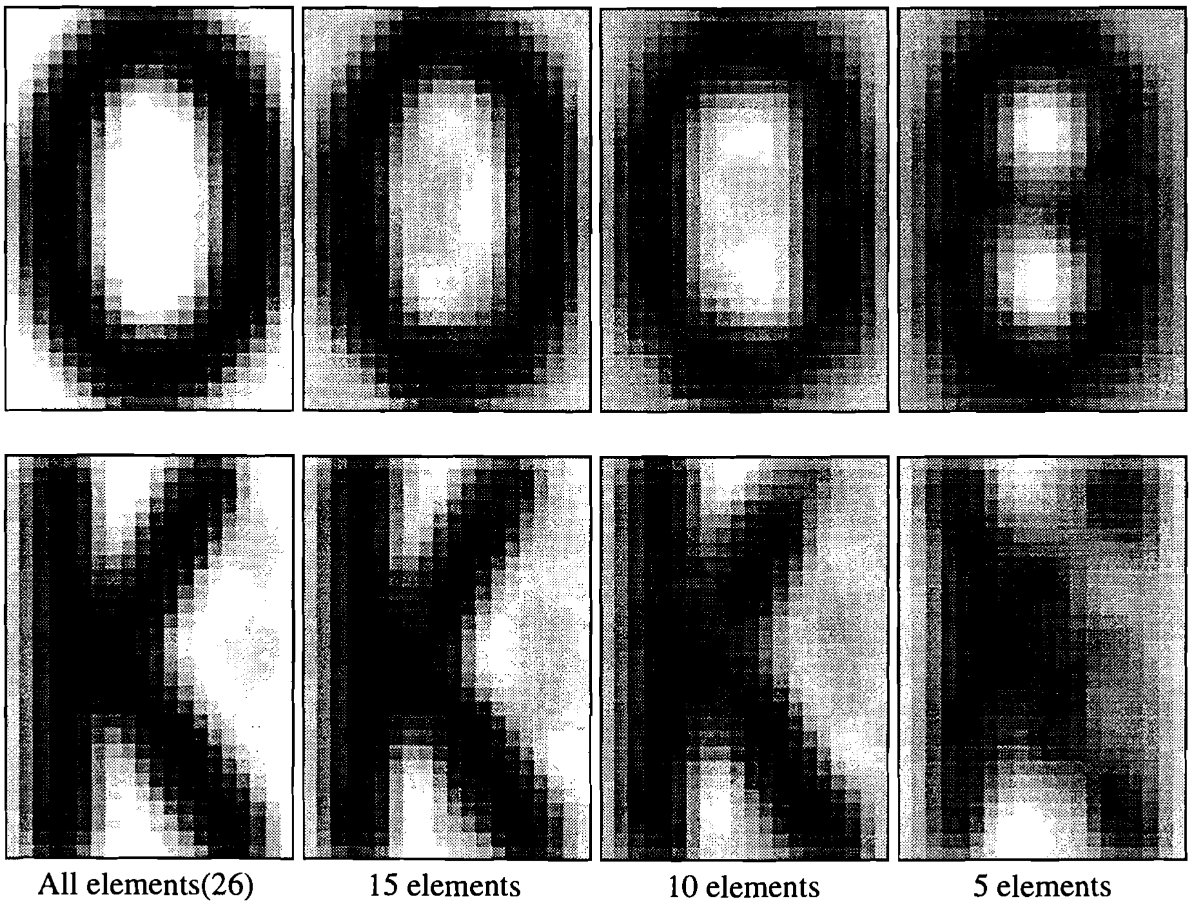


Figure 69. The reconstruction of the prototypes "0" and "K" when using all, 15, 10 and 5 elements of its PCA transformation.

Figure 69 shows that the reconstruction still looks very well if only 15 elements are used. The variation in the elements of the PCA transforms of a set of distorted characters is shown in figure 70.

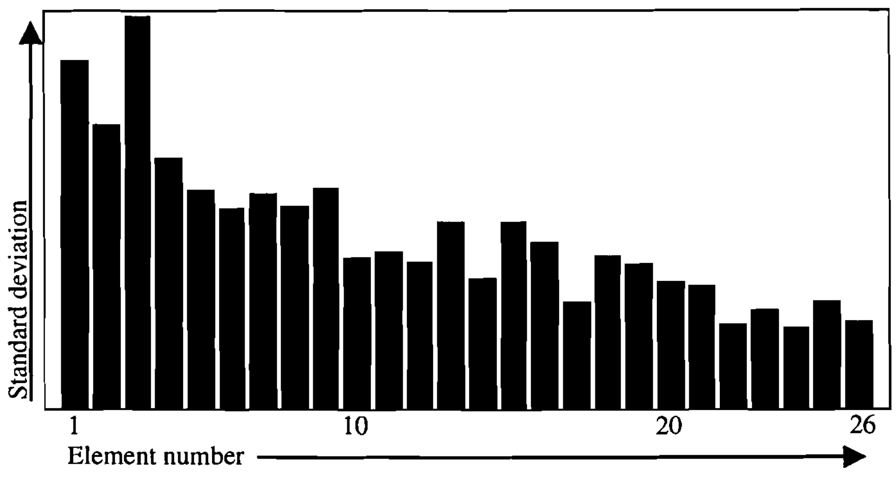


Figure 70. The standard deviation of the elements of a set of PCA transformed characters.

Figure 70 shows that the variation in the consecutive elements of a set of PCA transformed characters also tends to decrease. When looking at these figures the question arises if all elements are really needed to classify a PCA transformed character. Perhaps the last elements of a PCA vector contain more noise than information. The generalised Fisher's criterion [X] in equation (31) gives an estimation of the information, the average ability of discrimination for each component of a PCA vector.

$$D = \frac{\overline{\mu^2} - \mu^2}{\overline{\sigma^2}} \tag{31}$$

With μ^2 the square mean value of the considered element computed over all classes, and $\overline{\mu^2}$ and $\overline{\sigma^2}$ are respectively the average values of the square of its mean value and of its variance computed over each class. At this time all elements of a PCA transformed character are used.

If the variation in the consecutive elements of a PCA vector tends to decrease then one can wonder which shape of cloud a set of PCA transformed characters form. This was examined for some classes. It showed that the clouds of points are no spheres. A distribution like shown in figure 71 is more likely.

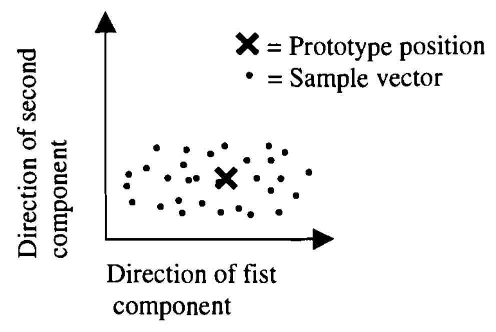


Figure 71. A possible distribution of PCA transformed characters of a class in 2 dimensions.

At this point one problem has to be solved, one has to find the most likely classification. In other words, one has to find a measure for the quality of recognition for each point in the PCA space. With this one can decide which quality is needed to accept a classification result. The quality of recognition is examined in the next paragraphs.

9.4 Measuring the quality of recognition

Finding a good measure for the quality of recognition is very important. Reducing misclassification depends on it. The problem of finding the areas that represent reliable recognition can be tackled in different ways. The starting point is a n-dimensional space with the prototypes indicated as points. The areas around the prototypes that define a sure classification have to be found for each prototype. It seems logical that the area close to a prototype represents a reliable recognition. So a distance measure or ratio can be used to define the area of secure classification.

9.4.1 The distance ratio

The Euclidean distance was tried to find the areas that represent secure recognition. The prototype closest to the transformed character is accepted then. The distance to the second closest prototype gives information about how sure the classification is. So the ratio between the distances to the closest and second closest prototype gives a measure for how secure the classification is. A small ratio indicates a sure classification, values close to 1 indicate insecure classification. Figure 72 shows an example of the resulting ratio in two dimensions for an imaginary set of prototypes. The dark areas are caused by the prototypes, they cause a small ratio. The white lines represent areas with a ratio of 1, they divide the space in areas that are closest to one prototype. A figure that divides a space like this is called a Voronoi tessellation [IX p. 225].

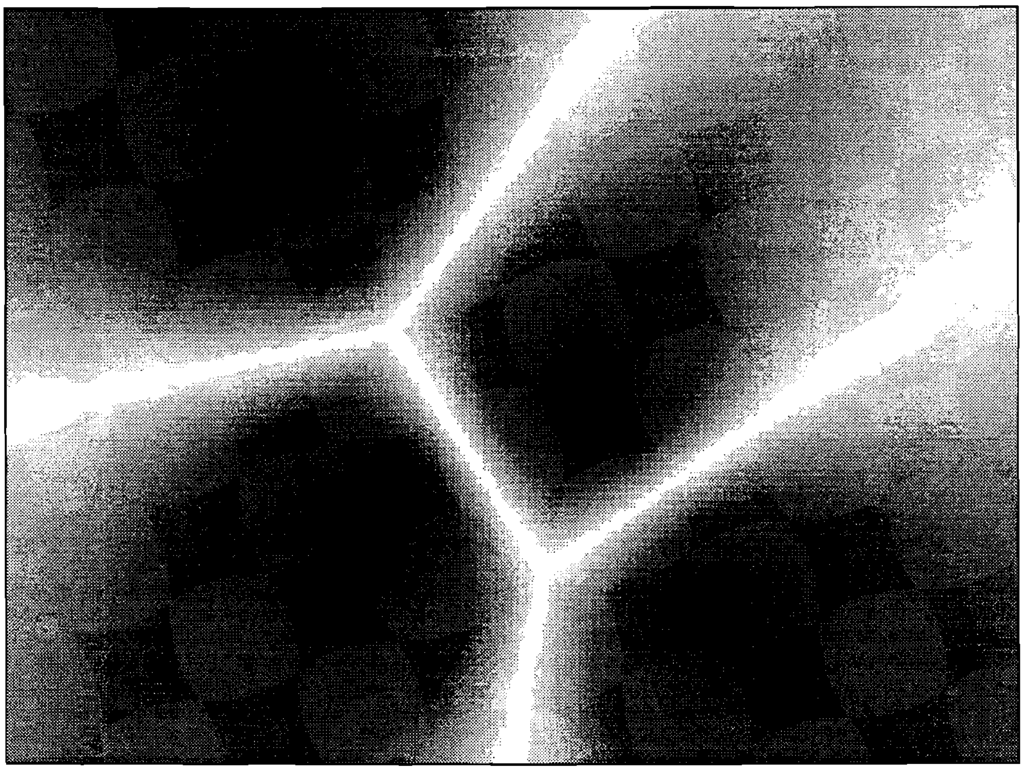


Figure 72. Ratio of the distance between the two closest prototypes.

The next figure shows some level lines of figure 72.

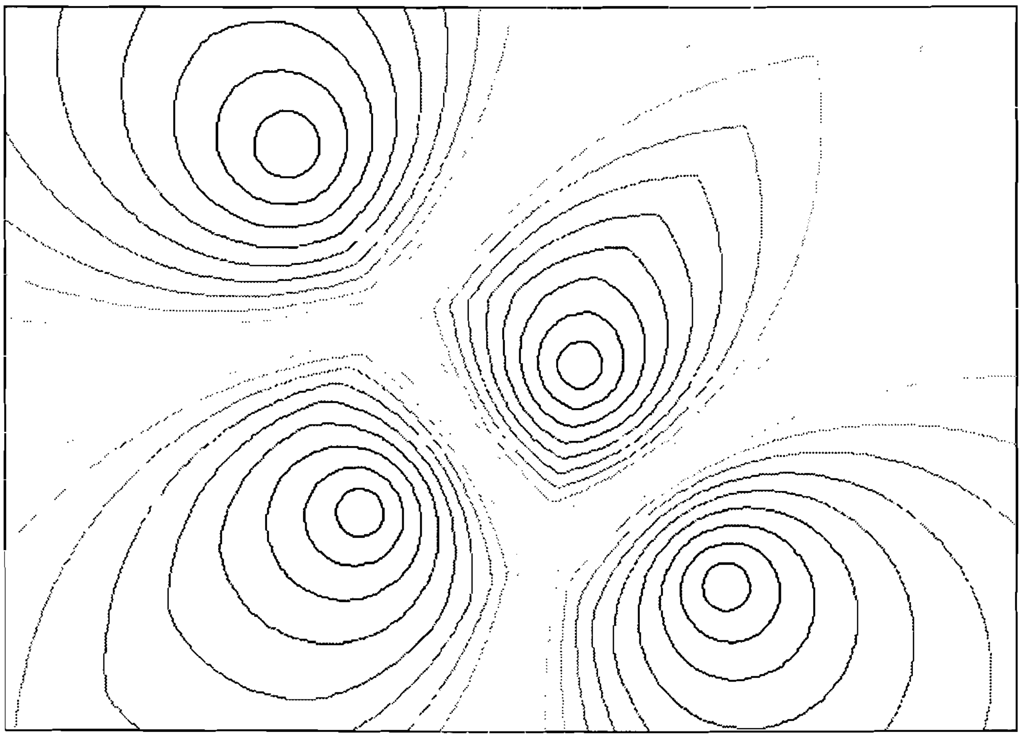


Figure 73. Level lines of the distance between the two closest prototypes.

A certain error and rejection rate belongs to each level line. The smaller the area in the level lines the smaller the error rate, but the rejection rate will increase. Another method to find a measure for the quality of recognition is described in the next paragraph.

9.4.2 The radial basis function network

A radial basis function network can also be used to find a measure of the quality of recognition. One can try to model the areas of secure recognition with a radial basis function and use a neural net to optimise the modelled areas. This can be done with a radial basis neural network like shown in figure 74.

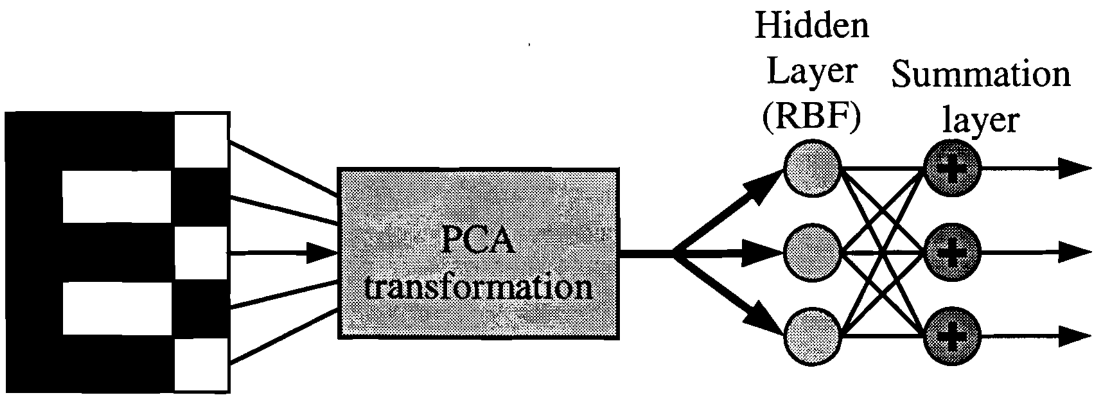


Figure 74. Classification with PCA transformation and a radial basis neural net.

The PCA vector can be classified with a radial basis neural network. A Gaussian function according equation (32) is used for this:

$$f(D) = e^{-\frac{D^2}{2\sigma^2}} \tag{32}$$

with D the Euclidean distance from a point to the PCA transformation of a prototype and σ the spread of the Gaussian function. Appropriate values for the sigma's have to be found. The values for the sigma's are not very critical [IX, p. 42-43]. The neural net will adapt to the chosen sigma's. A sigma close to zero will result in a nearest-neighbour classifier, so sigma should not be very small or very large. The sigma of each neuron is set to half the Euclidean distance to the closest neuron. This will lead to sigma's like shown in figure 75.

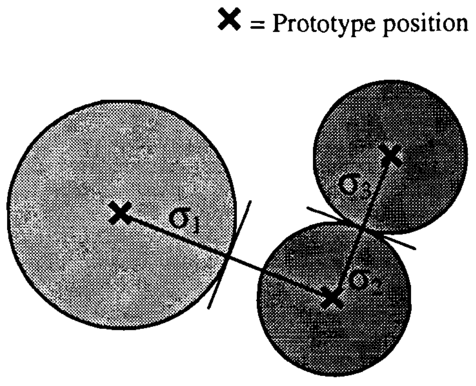


Figure 75. The sigma's of three neurons, $\sigma_1 > \sigma_2$ and $\sigma_2 = \sigma_3$.

This corresponds with the idea that points close to the PCA transformation of the prototypes are reliable. If the distance to the closest neuron is large then also the receptive field can be large. Neurons that lie close to each other indicate that the corresponding prototypes look alike. Therefore, their receptive field should be smaller.
No training is needed for this net, the optimum least mean square solution can be calculated like described in paragraph 3.2.1. The hidden layer proved to be a reliable source of information to draw conclusions about the quality of recognition. This is because the smooth transition in the response of one neuron to the next, like shown in figure 76.

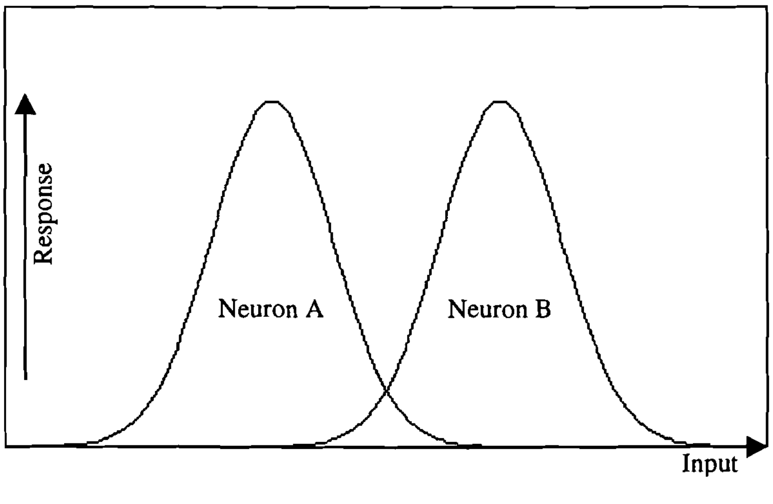


Figure 76. The response of two neurons in 1 dimension.

The area that one wants to accept or reject can be pointed out in this figure. The difference between the highest response and the runner up can be used for this. No Gaussian radial basis function is needed for this criterion. Just a distance measure would suffice. This method would even be equal to the method that uses the distance ratio if all sigma's had the same value. Figure 77 shows an example of the difference in response between the highest and second highest response in two dimensions for an imaginary set of prototypes.

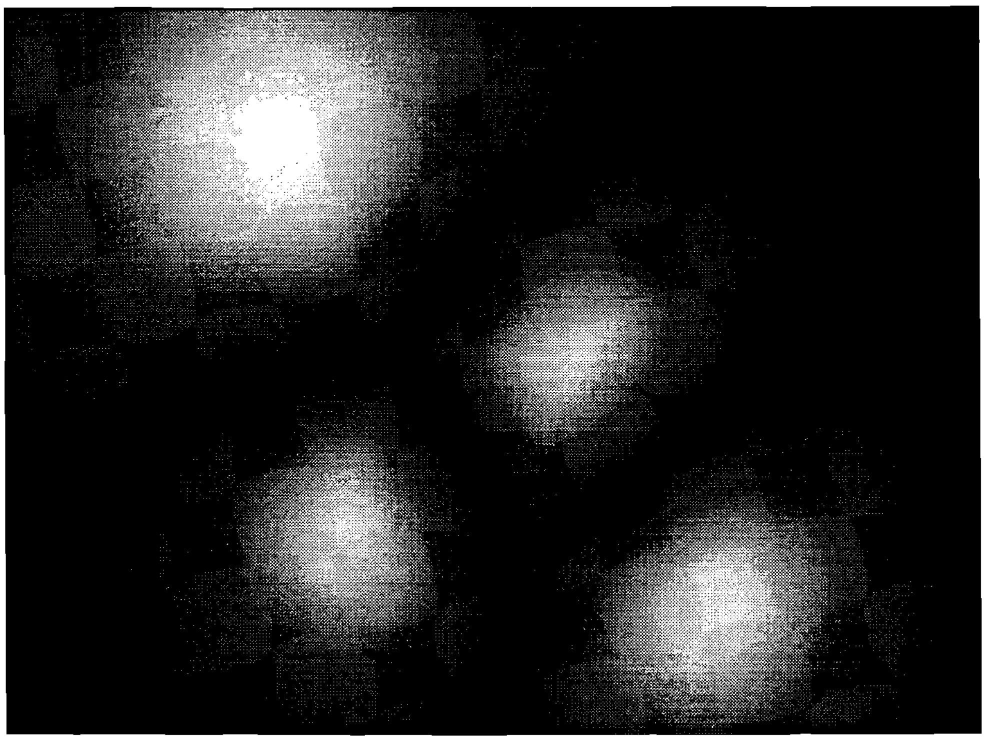


Figure 77. The difference between the highest and second highest response of the RBF neurons.

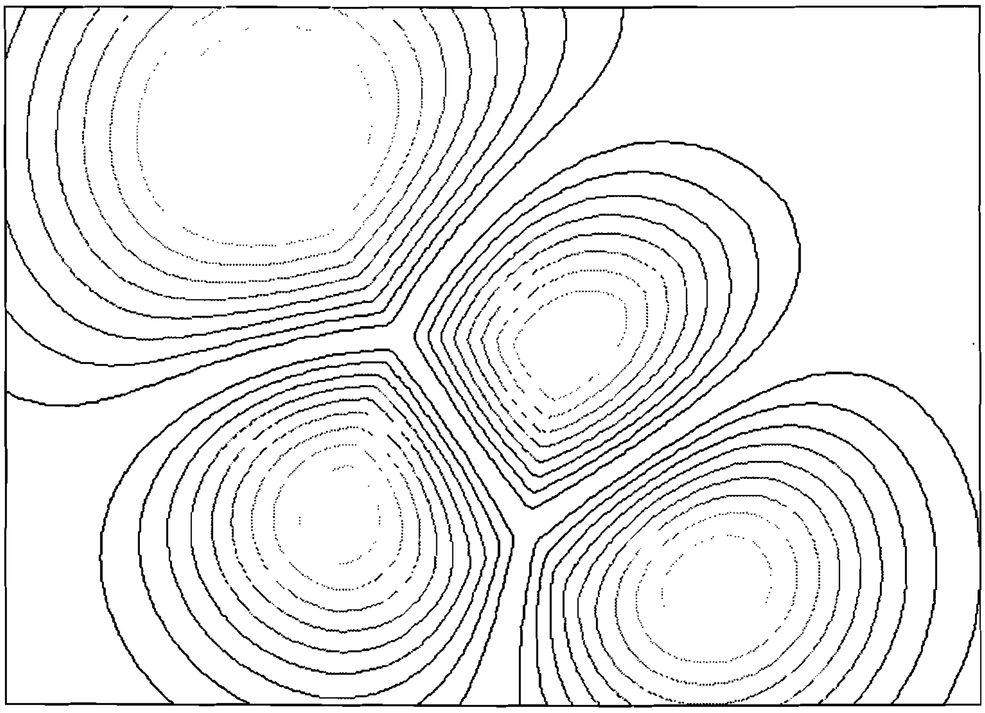


Figure 78. Some level lines of the difference between the highest and second highest response of the RBF neurons.

These modelled areas can now be refined with the help of the neural net. If one examines the neural net closely then one can predict the problems that will occur after training. The net is trained to respond with a 1 at one output and with a 0 at all others outputs. The problem is caused by the fact that the net tries to achieve this for all sample vectors, independently if the sample vector represents a good or a bad character. This is not what one wants because this does not give any information anymore about how certain the net is about a classification result. So the net points out the areas where an output should be 1 and all other zero, but not the areas where it is insecure. Two points that lie very close to each other will both be recognised as if they represent a very sure classification. This is illustrated in figure 76.

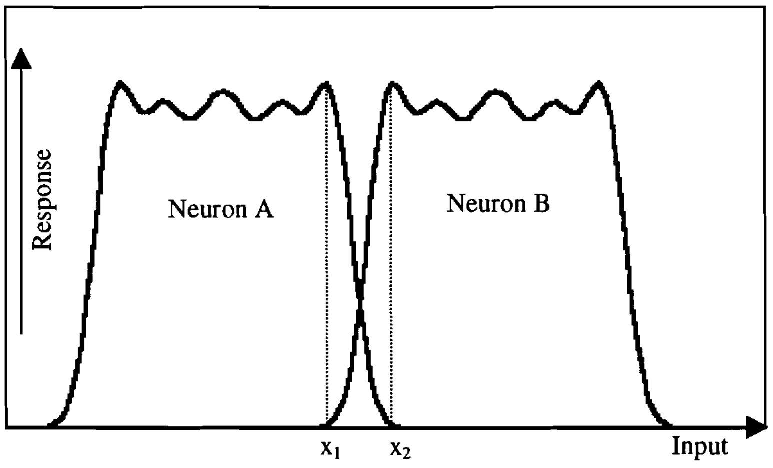


Figure 79. The response of two neurons of a "trained" neural net.

So applying a neural net in this way offers no improvement. In this approach it is tried to change the form of the receptive field at the output of the neurons with the help of a neural network. Perhaps the neural net should be put at the inputs of the RBF neurons. This would result in equation (33).

$$Y_i = e^{\frac{\|W_i \underline{x} - p_i\|^2}{2\sigma_i^2}} \qquad (33)$$

with Y_i the response of a neuron representing the prototype p_i on the input vector $\underline{x}$. W_i is the neural net for the neuron. It scales and rotates the axes of $\underline{x}$ to make the receptive field of the neuron optimal. However also with this variation more information than just the right classification result for each character is needed to train the net.
Perhaps this problem can be solved with a Bayesian classifier, which is examined in the next paragraph.

9.4.3 The Bayesian classifier

The basics of a Bayesian classifier were already discussed in paragraph 3.2.2 about probabilistic neural nets. Half of the sample data of the set of good plates was used to initialise the neural net. How these PCA transformed characters are collected is very important for the performance of the Bayesian classifier. One of the transformed characters coming from shifted characters can be used, but which one? The selection will influence the density and spread of the PCA transformations that represent one class. The vectors were selected with the help of the radial basis function described earlier. The PCA transformed character with the highest response was used.

A good value for the sigma is important for this stage. A large sigma value will cause most common characters to be recognised too often. A small value leads to a nearest-neighbour classification. No values for the sigma's were found that performed better than the approach with the distance ratio. A Bayesian classier should perform best for this kind of problem, but the amount of training data needed is probably much more than used here.

9.5 Syntax checking

A syntax check stage can be used to filter out impossible classification results. There are many rules for the syntax of the most common Dutch license plate model. Only the two most important rules are used here:

- Each pair of characters consists of two digits or two letters of the alphabet.
- There are at least two digits or two letters of the alphabet in a plate.

Actually there are many more rules that can help in finding impossible classification results. For example some character combinations are not used.
Here the syntax checker is used for two things, first of all to check the syntax, but also to try to find the correct syntax if an error is detected. To do this all character pairs are checked. If a character pair is found that contains a digit and a letter of the alphabet then the quality of recognition is examined for these two characters. If one character is recognised with a high quality and the other with a low quality then the character with a low quality is probably misclassified. The character is then classified again with the a-priori knowledge that it is a digit or a letter of the alphabet. Also if a valid syntax if found then the characters are classified again with the a-priori knowledge which pairs are digits and which are letters of the alphabet. This leads to an increase of the recognition quality and thus to a lower rejection rate because not all prototypes have to be taken into account. After the syntax check the final results are available.

9.6 System overview

The whole scheme of the license plate recognition system is presented in figure 80. It roughly shows the major steps of the system.

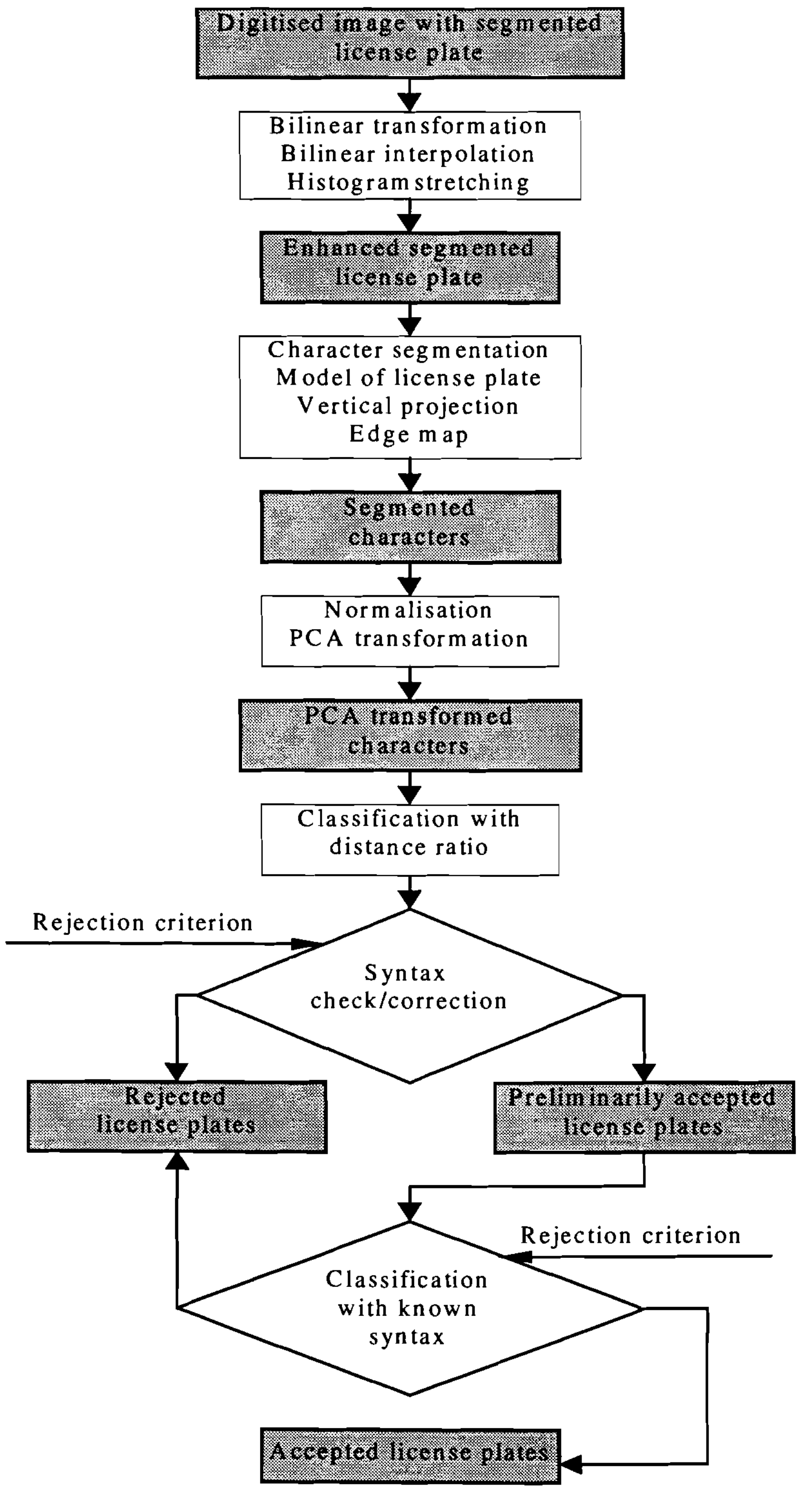


Figure 80. The scheme of the license plate recognition system.

9.7 Conclusions and recommendations

The distance measure performs best in terms of rejection rate and misclassification rate. This approach probably does not represent the optimal solution for this problem. Probably the Bayesian classifier performs better if more training data is available.

The applied method for the segmentation error correction seems to have much potential. Even segmentation-free classification seems to be possible with this approach. Here the six positions with the lowest distance measure are examined further. The actually positions that these six positions represent are not examined. These positions should correspond to a logical structure of positions. If for example in one corner a low distance is found, but the neighbours represent a high distance measure then the position in the corner is probably not optimal segmented.

The used white space around the prototypes is very important for the distance measure. These areas do not matter for the PCA transformation. The size and scaling of prototypes and the white space around them is probably not optimal at this moment.

The probability of a certain character for each character position in the license is not equal. This is caused by the rules that are applied for license plate numbers. Even certain character combinations occur more often than other combinations. At this time no use of these properties is made. Using them could improve the system performance.

Many rules that could help to detect misclassification are not yet used by the syntax checker. The syntax checker should be extended with these rules.

In a previous project [I] the scaling and segmentation of a character was adapted based on statistical properties of the character. Here this is not used because the distortions are here different than in the described project. Perhaps a experiment can show if this assumption is right.

10. The implementation

All the algorithms described in the previous chapters were implemented in Matlab 5.0. The next paragraph gives some general information about Matlab.

10.1 Matlab 5.0

Matlab is an abbreviation for Matrix Laboratory. It is an elaborate object oriented software package for scientific numeral calculations. Support, information and demos are available at the Internet http://www.mathworks.com.
Matlab is available for many different operating systems. Mainly Matlab 5.0 for Microsoft Windows 95 was used to develop the algorithms for the license plate recognition system. Also version 5.0 for SUN OS, which is a Unix like operating system, was used. These different versions should be fully compatible according the specifications of Matlab but experiments proved that they are not. The fact that SUN OS is case sensitive and Windows 95 is only case aware causes one difference. In Matlab for SUN OS all character in the file names are assumed to be lowercase. If under SUN OS a filename for a function or a script contains uppercase character then Matlab will not find it. A batch script named "lowercas" was written for SUN OS to rename all files in the working directory to lowercase. This will solve this problem.
Matlab consists of a basis that can be expanded with toolboxes. A toolbox consists of a number of functions, scripts and data, each stored in a separate file. It is necessary to have certain toolboxes installed before a script of a function will work. The programs written for the license plate system only need the image toolbox installed. All other functions used are part of the Matlab basis system. The image toolbox contains functions to read, write and manipulate images.
Matlab has some properties that can lead to problems in portability of a program. It is allowed to declare a variable with a name of an already existing function like for example "sin" in Matlab. If this is done then the function will become inaccessible.
The search path of Matlab can also cause some problems. The search path tells Matlab where to look for functions and scripts. The working directory is always searched before the directories in the search path. This becomes a problem if one for example changes to working directory to a backup directory to have a look at a previous version of a function and then continues experimenting. The programs will then use old versions of functions and read old data, so it is needed to make sure that the working directory is the right one.
A large number of toolboxes containing many functions is available in Matlab. Often a function needed is already implemented. However, where can one find it? Most of the time the built-in help function cannot answer this question. The help about a function usually refers to other functions about the same subject, but the references are far from complete. For

example, the help about the function "plot" does not refer to the function "stem" although they are both plot functions. Usually no references to functions in toolboxes that do not belong to the basis system are mentioned in the help function.
Another drawback of Matlab is the way it stores its data. The only data type that allows a reasonable freedom of manipulation is the "double array", which is eight bytes large. If one for example wants to store and manipulate a black and white image of 3000 × 2000 pixels then the amount of $24E^6$ bytes is needed. This is very much considering that the image just holds $75E^4$ bytes of pixel data.
Nevertheless, Matlab is a nice program to experiment with in spite of the drawbacks. It contains very powerful functions that can be applied to two or even higher dimensional data. Matlab is ideal for rapid prototyping, which is very important for this project. At the end the methods used are important, not the implementation. The implementation is important for the developers of the system. It tells the exact details about the used methods. The implementation will be discussed in the next paragraphs.

10.2 The implementation

The implementation consists of many functions and script files. Some general-purpose functions were written. They are roughly described here.

- FRAME. This function removes all columns and rows of a certain value (colour) from the borders of a matrix. It is used to strip white space around characters.
- EXPAND. Does the opposite of FRAME. It adds borders around characters.
- FEXIST. Checks if a given filename exists on disk.
- TO_ALFA. Gives the character belonging to its index, or the index belonging to the character.
- SHOWPROT. This function shows the prototypes.
- SHOWCOMP. This function shows the PCA components.
- DISTANCE. Calculates the Euclidean distance between two vectors.
- SDISTANC. Does the same as DISTANCE, but returns the squared distance.
- CALCSIGM. Calculates the sigma values for the RBF neurons. Half the distance from the neuron to the closest distance is calculated. Two times the squared sigma is returned.
- LP_PATH. Returns the path to the license plate toolbox. This file has to be adapted after installation.
- PLATEPAT. Returns the path to the license plate images. This file has to be adapted after installation.
- FIGURE?. The "?" stands for "1","12","21","22",23","32","24", "42" or "26'. This is a group of functions which are used to open windows at specified locations on screen.
- RADIAL. Calculates the response of the hidden and output layer of a radial basis neural network.

- TRAINR4. This function trains the radial basis function neural network. It only focuses on the winner and runner up.
- SYNTAXCH. Checks if the results of the classification of a Dutch plate meet some syntax criteria.
- BAYES. Calculates the response of one class in a Bayesian network. It is used by BAYESCLA.
- BAYESCLA. This function calculates the response of a Bayesian network for all classes.
- SPLITBAY. Splits up a set of PCA vectors in different classes. This is needed for the Bayesian classifier.
- SPLITSET. Splits up a set of PCA vectors. This can be used to generate a train and test set.
- SHOW. Shows an image. The image is histogram stretched before showing it.
- NORMALIS. This function normalises the columns of a matrix so that the average of the columns becomes 0 and the standard deviation becomes 1.
- HISTRETC. Stretches a histogram as described in paragraph 7.1. Actually the pixel values are changed, not the colour map of the image.
- RESAMP. This function resamples a rectangular area in an image. It is used to scale characters to the wanted dimensions.
- MIN12. Finds the smallest and the second smallest value in a data set.
- MAX12. Finds the largest and the second largest value in a data set.
- BIRESAMP. This function transforms an area pointed out by four corner points with the bilinear transformation. It is used to extract a license plate for an image. A description can be found in paragraph 6.1 and further.
- PSEUDO_I. This function calculates the pseudo inverse of a matrix.
- W_INIT. This function calculates an initialisation matrix for the RBF network with the pseudo inverse.
- KLTRANSR. Calculates the Karhunen-Loève (PCA) transformation.
- EIGHANS. Calculates the sorted eigenvalues with their eigenvectors.
- SHOWSEGM. Shows the results of segmentation in an image.
- PCA_CLAS. Classifies a PCA vector with the distance ratio method.
- CHAR2PCA. Calculates the PCA transformations of the character in all positions.

These functions are called from the functions and scripts that form the body of the system. Some programs are implemented as scripts, this makes debugging them easier because then all data is still accessible after termination of the program. Most of the programs that form the body of the system are already discussed in the previous chapters. Here the rough outline starting from the prototypes to the results of classification is explained.

10.2.1 Calculating the prototypes

Prototypes form the basis for the PCA transformation. The quality of the prototypes is very important. The best way to calculate them is by taking the average of a large set of real life characters of a class. A 2-stage process is used to achieve this. A lot of images were available, these were segmented manually with the help of LP_SEGM. The characters are obtained by segmenting the characters of a selected group of plates automatically with the program GET_CHAR, which calls SEGMENT to segment the characters on the plates. All plates with heavily distorted characters or character segmentation errors were excluded from this group. The resulting 461 plates were used to calculate the prototypes.
The characters must be segmented exactly before one can take the average. This process can be automated. Some reasonably well segmented prototypes are needed for this. Some images of scanned pictures of plates were used to get these prototypes. Not all characters were found in these pictures. The missing characters were scanned from the official description of the license plate characters. The "Q" is not included in the official description, so no prototype is available of this character. Instead the dash is substituted. The resulting prototypes are stored in the image "PROTOTYP.BMP" which is shown in figure 81.

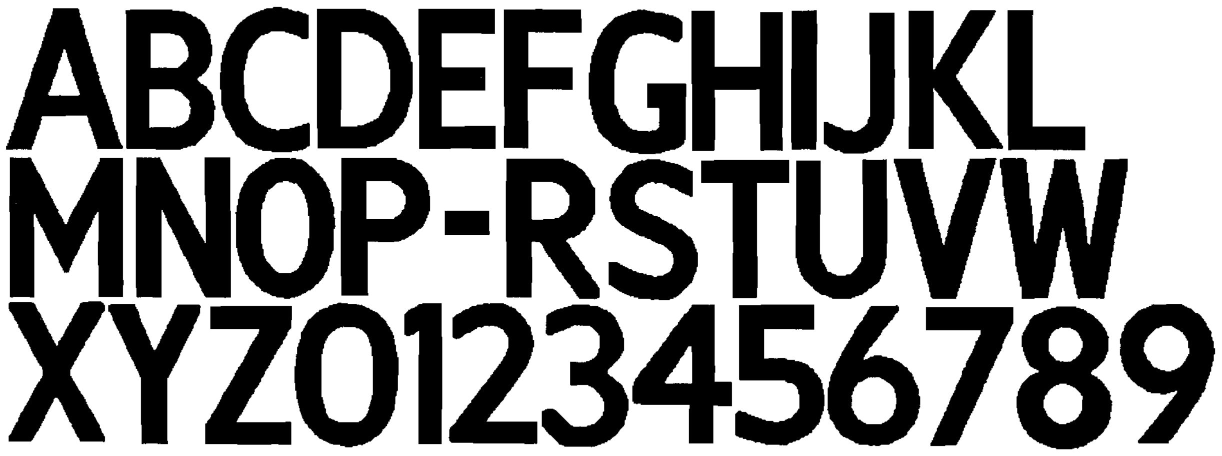


Figure 81. The initialisation prototypes.

These prototypes are segmented by PROT_SEG. The results are scaled to the wanted dimensions and put in a window of 28 x 20 pixels. Some white space is left around the prototypes. Two examples are shown in figure 82.

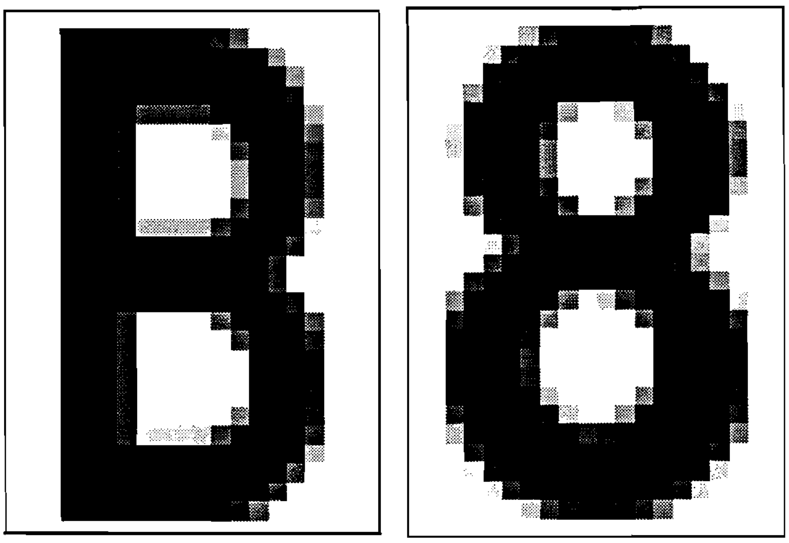


Figure 82. A scaled and segmented initialisation prototype “B” and “8”.

The approximately segmented characters can be classified with these prototypes. The classifier returns information about the segmentation correction. This is used to correct for segmentation errors. Then the prototypes can be calculated from the correctly segmented characters with the help of the function MAKEPROT, the results are stored in PROT_AVR.MAT. Two prototypes which were calculated this way are shown in figure 83.

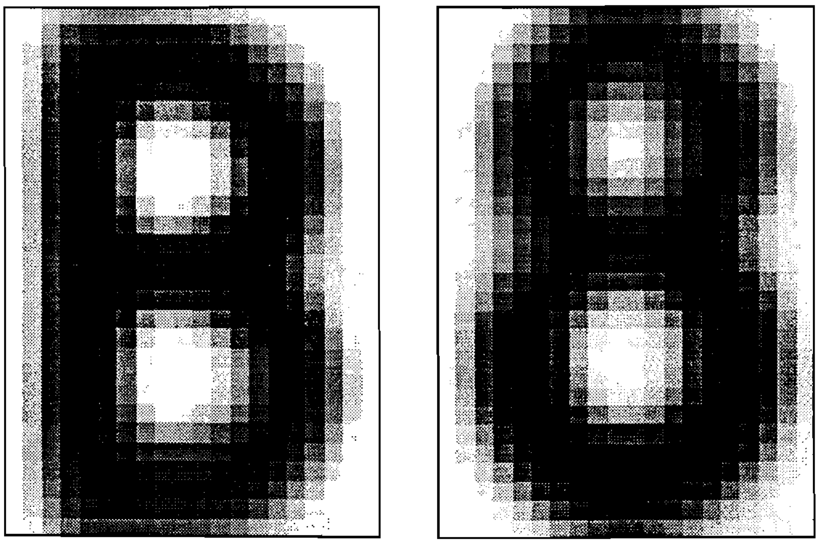


Figure 83. The prototype “B“ and “8”.

The PCA transformation is calculated from these prototypes with the CALC_ALL, the results are stored in PCA_TRAN.MAT. Then the PCA transformation with the distance measure from the original and the reconstruction of a character can be calculated for all characters. This is done with the help of the function PCA_DATA, the results are stored in PCA_DATA.MAT. The resulting data can be analysed with the different classification schemes. The results of the classification are checked by a syntax check stage. The whole trajectory is illustrated in figure 84.

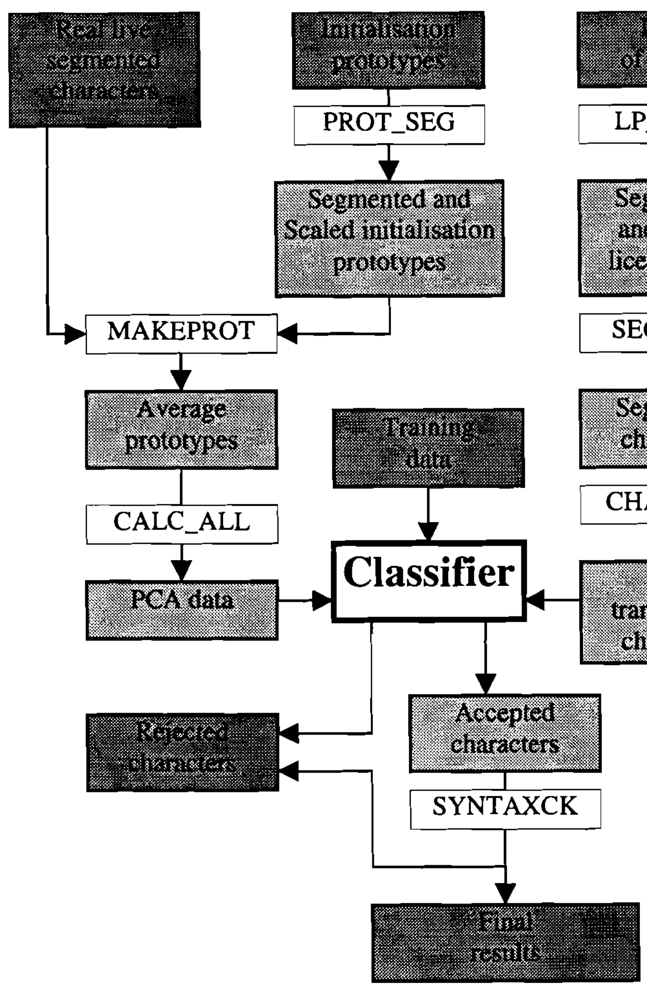


Figure 84. The scheme of the recognition system.

Different classification methods can be used, they differ in the way they classify the PCA vectors. RECOGN follows the scheme presented in figure 84.
All characters can be segmented and saved as separate files with GET_CHAR. Then the PCA data with the distance measure can be calculated with PCA_DATA which creates a files called PCA_DATA.MAT. This mat file can be analysed with ANAL_PCA.
The syntax of a plate is checked with the program SYNTAXCK. It only checks if all character pairs consist of two digits or two letters of the alphabet and if there are no more than four digits or four letters of the alphabet in the plate.

11. System evaluation

Here the recognition system is evaluated with the help of the results. 545 plates were used to evaluate the system.

The evaluation of the consecutive stages of the system:

- Camera and frame grabber;
- License plate segmentation;
- Character segmentation;
- Classification;
- Syntax checker;

will be treated in the following paragraphs.

11.1 The camera and frame grabber

The images obtained form the frame grabber can be very different. Some license plates in the images are very bright and some are very dark. The applied image enhancement and normalisation corrects for this with success. Special attention must be paid to a good setup of the camera and frame grabber. Properties like contrast, sharpness, noise, camera angle and used artificial lighting all influence the performance of the system.
The resolution of the characters delivered by the frame grabber proved to be enough for the approach described here. The character height is on average about 18 pixels. The recognition system would benefit from a better resolution.
At this time a monochrome camera is used. Using a colour camera could help in recognising foreign plates. In addition, the license plate segmentation stage can benefit from a colour camera.

11.2 The license plate segmentation

The license plate segmentation stage cannot be evaluated because it is unavailable to us. The rejection and error rate is unknown. All plates were segmented by hand, which resulted in an error and rejection rate of 0. It was tried to achieve a segmentation accuracy according to the specifications given by CFT. It would be best to test the system with automatically segmented plates to really see if the character segmentation and classification is able to deal with the errors that are introduced by the license and character segmentation stage.

The plates were extracted from the images with the bilinear transformation. This corrects for most of the perspective view distortion, but it cannot correct for distortions that are caused by bent plates. If the corner points of a plate are known then it should be possible to find the contours of the plate. A more advanced transformation can then be used to calculate a better reconstruction of the plates. This will result in a lower error and rejection rate.
Bilinear interpolation is used to resample the plates in the images to fixed dimensions. This introduces some blurring in the images, which actually enhances the images a little. However, the amount of blurring depends on the resolution and angle of the plate in the image. It is unknown if this causes degradation of the performance of the system.

11.3 The character segmentation stage

The character segmentation proved to function very reliable. Only 6 out of the 545 plates were segmented badly. They had large segmentation errors in the segmentation of one or more characters on the plate. All other characters are segmented with an accuracy of about 2 pixels horizontally and 1 pixel vertically to each direction. The license plates are enhanced for this stage with histogram stretching. This improves the reliability of the character segmentation.
The segmentation accuracy can be measured with the help of the classification stage. The classification stage applies a segmentation correction, the needed segmentation correction gives information about how accurate the initial character segmentation was. The segmentation error correction shifts the character up and down and sideways. It also tries if the classification result is better if the first and last row of the character are ignored. At first 7 horizontal and 5 vertical positions were tried. The percentage of the number of times that a position delivered the highest and the correct result is shown in table 2. All (3270) characters were treated as independent characters, no syntax checking was used. No characters were rejected. 3172 characters were classified correctly, 98 were misclassified.
A Gaussian function was used to measure the quality of recognition for the PCA transformed characters. The sigma's of the Gaussian functions were set to half the distance of the neuron to the closest other neuron.

Table 2. The percentage of the total number of correct winning results per position for 35 positions.

		Horizontal position						
		3 left	2 left	1 left	Basis	1 right	2 right	3 right
Vertical position	2 pixels up	0	0.1	0.2	0.2	0.4	0.2	0.2
	1 pixel up	0.1	2.8	6.1	6.0	6.7	3.0	0.4
	Basis position	0.1	2.3	6.5	8.1	7.2	3.1	0.1
	1 pixel down	0.1	2.7	5.5	5.2	7.1	3.0	0.2
	2 pixels down	0.2	0.1	0.1	0.3	0.1	0	0
	Without first and last row	0	1.8	5.3	6.3	6.1	2.1	0

The percentage of the number of times that a position delivered the winning, but wrong result is given in table 3.

Table 3. The percentage of the total number of misclassifications per position for 35 positions.

		Horizontal position						
		3 left	2 left	1 left	Basis	1 right	2 right	3 right
Vertical position	2 pixels up	1	1	1	3	2	3	2
	1 pixel up	1	2	3	7	4	2	1
	Basis position	3	3	3	10	4	2	0
	1 pixel down	2	2	2	6	1	3	2
	2 pixels down	0	0	2	3	3	2	2
	Without first and last row	1	2	2	1	2	3	1

Table 2 indicate that the segmentation error is indeed about 2 pixels vertically and 1 pixel horizontally. Shifting 3 pixels to each side and 2 pixels up and down does not seem to be needed for a correct result, these positions actually contribute relatively much to the misclassification. That is why they are not used anymore. Thus only 20 positions are taken into account.
At this stage also the segmentation was enhanced which resulted in a better horizontal segmentation but this also seems to have caused an offset in the horizontal and vertical segmentation. The enhanced segmentation reduced the number of misclassifications to 69. The enhanced segmentation seems to indicate that perhaps only three horizontal positions are need for an accurate horizontal segmentation. Table 4 shows the percentage of the number of correct winning results per positions.

Table 4. The percentage of the total number of correct results per position for 20 positions.

		Horizontal position				
		2 left	1 left	Basis	1 right	2 right
Vertical position	1 pixel up	0.52	2.9	16.2	13.7	1.0
	Basis position	0.58	3.6	15.7	10.4	0.92
	1 pixel down	0.61	2.8	9.4	6.2	0.52
	Without first and last row	0.12	1.8	7.5	5.1	0.37

There is an offset in the segmentation which was not further examined because of time constraints. In the optimum case there should be a maximum at the basis position.
The percentage of the number of times that a position delivered the winning, but wrong result is given in table 5.

Table 5. The percentage of the total number of misclassifications per position.

		Horizontal position				
		2 left	1 left	Basis	1 right	2 right
Vertical position	1 pixel up	1.4	8.7	10.1	14.5	2.9
	Basis position	0	2.9	20.3	7.2	0
	1 pixel down	0	1.4	5.8	5.8	0
	Without first and last row	0	5.8	8.7	2.9	1.4

The total number of misclassifications is 69, which is 2.1% of the total number of characters. The results with the reduced number of positions are better. The number of misclassifications is lower. This is because some errors are caused by shifting the characters to far, which can lead to misclassification if the bad segmentation is not detected.

The classification stage is able to deal with the errors introduced by the segmentation stages, but improving the segmentation will result in a lower misclassification and rejection rate.

11.4 The classification stage

Here the different classification methods are compared with each other. All methods are based on the PCA transformation of the characters. The methods actually only differ in the way they analyse the PCA transformed characters.

11.4.1 Distance ratio

Here the distance to the closest prototype is used to classify the characters. The ratio between the closest and second closest prototype is used to decide if the character classification is accepted or rejected. One minus this ratio can be interpreted as a threshold to reject or accept a result. If the threshold is set to 0, thus all character are accepted, then 63 out of the 3270 characters are misclassified, which is 1.9% of the characters. Figure 85 shows the error rate as a function of the rejection rate.

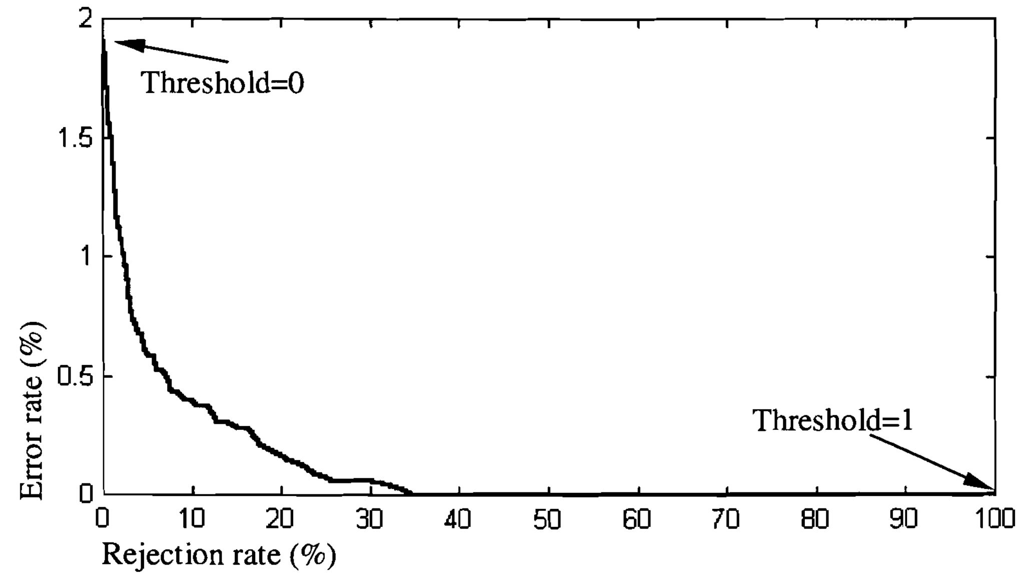


Figure 85. The error rate as a function of the rejection rate for the distance ratio method.

Figure 86 shows the threshold as a function of the rejection rate for the distance ratio method. The shown graph can be used in combination with figure 85 to find the threshold at a certain error and rejection rate.

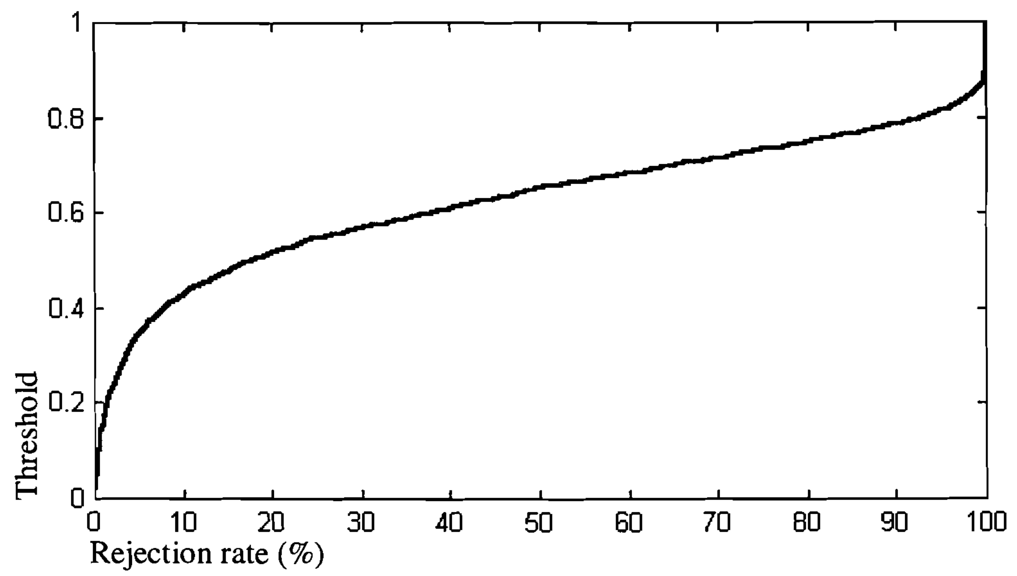


Figure 86. The threshold as a function of the rejection rate for the distance ratio method.

11.4.2 The RBF layer

In this method a RBF layer is used to classify the characters. A Gaussian function is used for this. The sigma's of the RBF neurons are set to half the distance of the neuron to the closest other neuron. The RBF neuron with the highest response is accepted as the classification result. The difference between the highest and second highest response is used as threshold to accept or reject a result. The error rate as a function of the rejection rate is shown in figure 87. 69 classification errors were made at a threshold of 0.

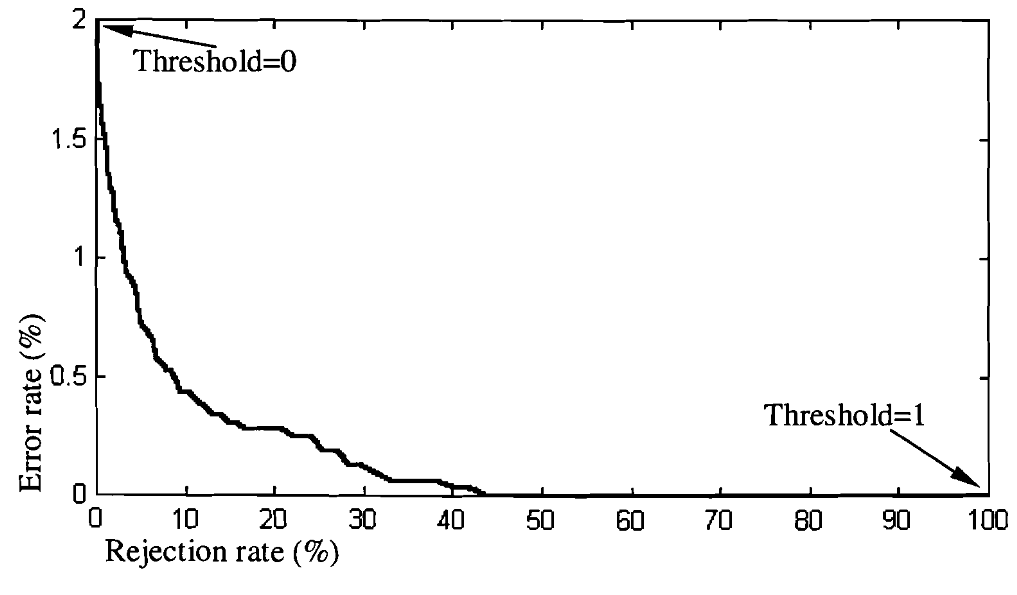


Figure 87. The error rate as a function of the rejection rate for the RBF layer method.

Figure 88 shows the threshold as a function of the rejection rate for the RBF method. The shown graph can be used in combination with figure 87 to find the threshold at a certain error and rejection rate.

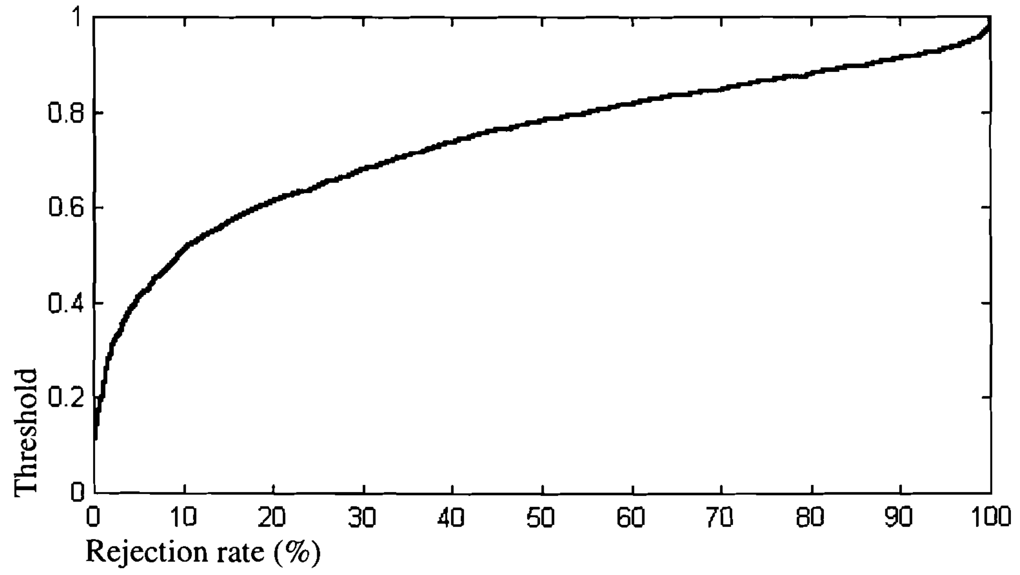


Figure 88. The threshold as a function of the rejection rate for the RBF layer method.

In the next figure the results of the classifier with the distance ratio and the RBF layer can be compared.

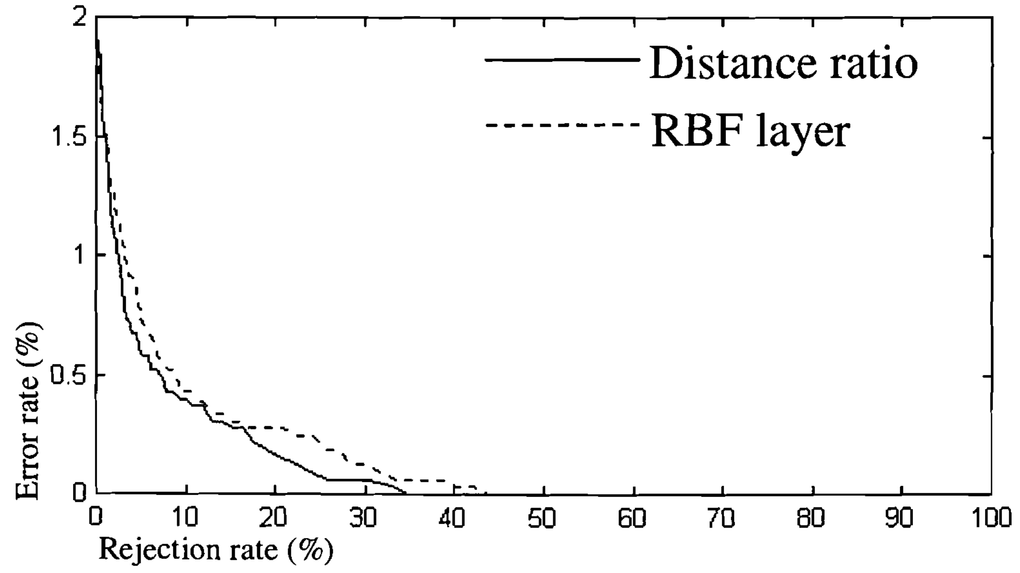


Figure 89. The error rate as a function of the error rate for the RBF and ratio approach.

The figure indicates that the approach that uses the distance ratio performs better than the RBF approach. A lower rejection rate at the same error rate as with the RBF layer can be achieved with it.

11.4.3 The trained RBF network

Half of the characters of the good set are used to calculate the optimum weight matrix, in respect to the mean square error, for the net. The classification results with the trained RBF network are about the same as with the RBF layer only. But the net does not give clear information about how sure the net is about the classification as predicted in paragraph 9.4.2. Using a trained RBF network in this way is not suitable for this problem.

11.4.4 The Bayesian classifier

Half of the characters of the good set are used to initialise the network. The misclassification rate of the Bayesian classifier is very dependent on the chosen sigma and the quality and amount of the training data. Especially the amount of training data causes the classifier to perform worse than the other classification methods. Measuring the quality of recognition proved to be hard for this classifier. Further examination is of this type of classifier is needed. A sigma value of 10 seemed to be an appropriate.

11.4.5 Conclusions

The classification approach that uses the distance ration proved to perform best in this application. 1 minus the distance ratio is interpreted as a threshold that regulates the misclassification and rejection ratio. This method is used for the time being. Table 6 shows the confusion matrix for a threshold of 0.35. At this point the rejection rate is 5.2% and the error rate is 0.6%. The characters are classified as individual characters, not as part of a syntax of a license plates. Only the used prototypes are shown in the table.

Table 6. The confusion matrix for a threshold of 0.35, 1 represents 0.03% of the total number of characters.

	B	D	F	G	H	J	K	L	N	P	R	S	T	V	X	Y	Z	0	1	2	3	4	5	6	7	8	9
B	-																									4	
D	1	-																2									
F			-																								
G				-																							
H					-																						
J						-													2								
K						1	-																				
L								-											1								
N									-																		
P			1							-																	
R											-																
S												-															
T													-						1								
V														-		1											
X															-												
Y																-											
Z																	-										
0		1																-									
1							1												-								
2																				-							
3																					-						
4																						-					
5																							-				
6																								-			
7																									-		
8	3																									-	
9																											-

Most of the misclassifications can be detected if the syntax of the plate is used, this done in the syntax check stage.

11.5 The syntax checker

The characters were treated as individual characters in the previous paragraphs. Here the syntax of the classification results of a plate is checked.
The role of the syntax checker is very dependent on the errors that the classification makes, and thus on the threshold that is used to accept characters. Almost all misclassifications are digits that are classified as a letter of the alphabet or the other way around at the threshold used here. If such errors do not occur two times in one pair of characters then the syntax checker will detect them and reject the plate. The syntax checker also checks if there are to many digits or letters of the alphabet in the plate. 15 errors were detected by the syntax checker.

12. Conclusions and recommendations

First of all, one can conclude from the results that using the PCA transformation for license plate character recognition is very well possible. A recognition rate of 87% with a misclassification rate of 0.4% was achieved with it. A compromise between the misclassification and rejection rate can be made by adapting the threshold. Further developing the Bayesian classifier can probably increase the performance still.

It is best if the license plate segmentation stage detects what kind of plate is present in an image. In addition, foreign plates should be detected there. The aspect ratio and colour of a plate give clear clues about the type of plate. If the type of plate is known then the enhancement, segmentation and classification can be adapted to it.

The character segmentation scheme based on the vertical projection performs good enough for the classification. The error rate is just 1.1%, which shows that it is possible to deal with the character segmentation problem in this way. However, a drawback of this approach is that it has to be redesigned for the different models of the Dutch license plates and foreign license plates. Segmentation should not be done independently of the classification, especially if the classification stage is able to correct for small segmentation errors. A co-operation of the classification and segmentation stage is most likely to perform very well. One can for example imagine a scheme where a segmented character is classified. The classifier then enhances the segmentation or even rejects it. This information is fed back to the segmentation stage, which responds accordingly.

In this project the license plate segmentation was done by hand. This leads to a lower rejection and error rate than with the automated system. Furthermore the license plate segmentation is probably more accurate when done by hand. To evaluate the system it would be best to have the automated system available.

To detect misclassification one can check the character spacing found by the classifier. This spacing has to obey to strict rules. If the characters are classified then the spacing is known and can be checked. Misclassifications caused by bad segmentation can be detected this way.

The applied segmentation correction is working very well. It is based on the properties of the reconstruction of a PCA transformed character. The difference between the original and the reconstructed character gives clues about possible segmentation errors. Segmentation-free classification even seems possible with this approach. All possible character area's on a plate can be PCA transformed and reconstructed. The differences between the original and reconstruction together with the response of for example a radial basis layer, which evaluates the PCA transformations, give information about the segmentation of the character.

There seems to be an offset in the vertical segmentation of the characters, this is concluded from the results of the system evaluation. Removing this offset will probably cause an increase in the system performance.

The used white space around the prototypes is very important for the distance measure. These areas do not matter for the PCA transformation. The size and scaling of the prototypes and the white space around them is probably not optimal at this moment. Optimising the scaling and white space will probably cause an increase of the system performance.

Acknowledgements

I have to thank my coaches Hans Hegt and Nadeem Khan. They always made time for me if technical of moral support was needed. Together we formed a team, which made working on this project very enjoyable. Both had the best intentions with me and even tried to help me ahead on my way into a professional career. I am definitely certain that such good coaches are very rare and consider myself lucky to have worked with them.
I also have to thank my girlfriend Ester, who in the course of time probably thinks that I love my computer more than her. She is wrong of course.
To close I'm much obliged to my friends and colleagues at the computer centre of the university for putting up with my boring stories about character recognition. Cheers to you all.

Bibliography

[I] Bogaards, H.
"Karakterherkenning met principal component analysis", department of Electrical Engineering, Electronic Circuit Design Group, Eindhoven University of Technology, 1997, project report.

[II] Chen, C.H.
"Fuzzy logic and neural network handbook", McGraw-Hill, 1996, ISBN 0-07-011189-8.

[III] Gonzalez, R.C & R.E. Woods.
"Digital image processing", first edition, Addison-Wesley Publishing Company, 1993, ISBN 0-201-50803-6.

[IV] Hegt, J.A.
"Neurale Netwerken", department of Electrical Engineering, Electronic Circuit Design Group, Einhoven University of Technology 1994, syllabus.

[V] Hertz, J. & A. Krogh & R.G. Palmer.
"Introduction to the theory of neural computation", Addison-Wesley Publishing Company, 1991, ISBN 0-201-50395-6.

[VI] Kreizig, E.
"Advanced engineering mathematics", 6th edition, John Wiley & Sons, 1988, ISBN 0-471-85824-2.

[VII] Centrum for Fabrication Technology
"Kenteken Verwerking (KEVER)", Philips Industrial Vision, design documentation, 1993.

[VIII] Vliet, R.G. Van
"Beeldbewerking", edition 1993, department of electrical Engineering, Control group, Eindhoven University of Technology , Syllabus.

[IX] Wasserman, Philip D.
"Advanced methods in neural computing", Van Nostrand Reinhold, 1993, ISBN 0-442-00461-3.

[X] Fukunaga, B.
"Introduction to statistical Pattern Recognition", Academic Press, London, 1990.

[XI] Nijhuis J.A.G. & M.H. Ter Brugge & K.A. Helmholt & J.P.W. Pluim & L. Spaanenburg & R.S. Venema & M.A. Westenberg.
"Car license plate recogniton with neural networks and fuzzy logic", Proceedings of the 1995 IEEE international conference on neural networks, 27 november-1 december 1995, Vol. 5, p. 2232-2236.