Final Project: Automated License Plate Recognition

by Nicole Ketelaars

Suppose you are on holiday in Italy. Suppose you visit the beautiful city of Bologna. Suppose you drive your car into the city center. Then you shouldn't be surprised to see a letter with a big fine in your mailbox, when you come back home. In Bologna, and several other Italian cities, the city counsel decided that cars are not allowed in the city-center. To make sure that all transgressors are caught in the act, they asked the Dutch company Beltech to develop an automated license plate recognition system.

I started my final project in January 2001 at Beltech. By then, the automated license plate recognition system (ALPS) was already in place for a couple of years. It was ready for a facelift. I looked at the image analysis problem from a different angle and started anew with designing a solution. The final result was successful, and it is now used in practice in combination with the first version of ALPS.

Figure 1: A typical photo of an Italian car.

Let's first have a look at the type of pictures that we received from the Italian police. A typical example of such a photo is shown in Figure 1. The existing system mainly had problems with finding the characters in a license plate where the characters seemed "attached" to the black bar directly above the license plate. A second bottle-neck was the difficulty to distinguish between characters that are very much alike, such as: **B** and **8**, **S** and **5**, **V** and **Y**, **D** and **0**, **G** and **C**.

We divided the problem in the following four consecutive steps:

1. Find the license plate on the photo (the location phase)
2. Find the "characters bitmaps" on the license plate (the segmentation phase)
3. Clean the character bitmaps, such that only the character remains (the cleaning phase)
4. Determine which character is displayed in the character bitmap (the classification phase)

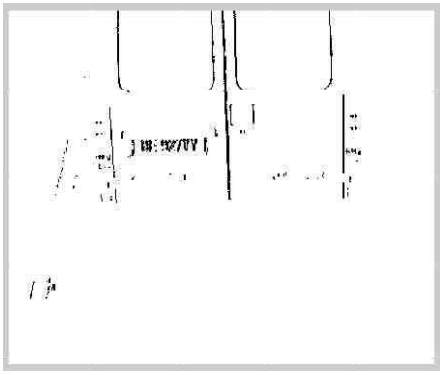

Figure 2: De edge-image relating to the Figure 1.

The Location Phase

In the location phase, we have as input a greyscale photo and we want as output the part of the photo on which the license plate is displayed. For us, human observers, this is an easy assignment. We can all point to the license plate region. For a computer, however, the photo is just a matrix of values between 1 and 255 (representing grey levels). The structures that we see in a picture have to be caught in algorithms or rules and given as input to the computer, before the computer can find these structures.

The algorithm for the location of the license plate is based on the following idea: a license plate consists of a row of characters. A character is a "relatively dark thing on a relatively light background". So, we can say that the license plate region is characterised by a row of transitions from dark to light and vice versa (we call such a transition an edge).

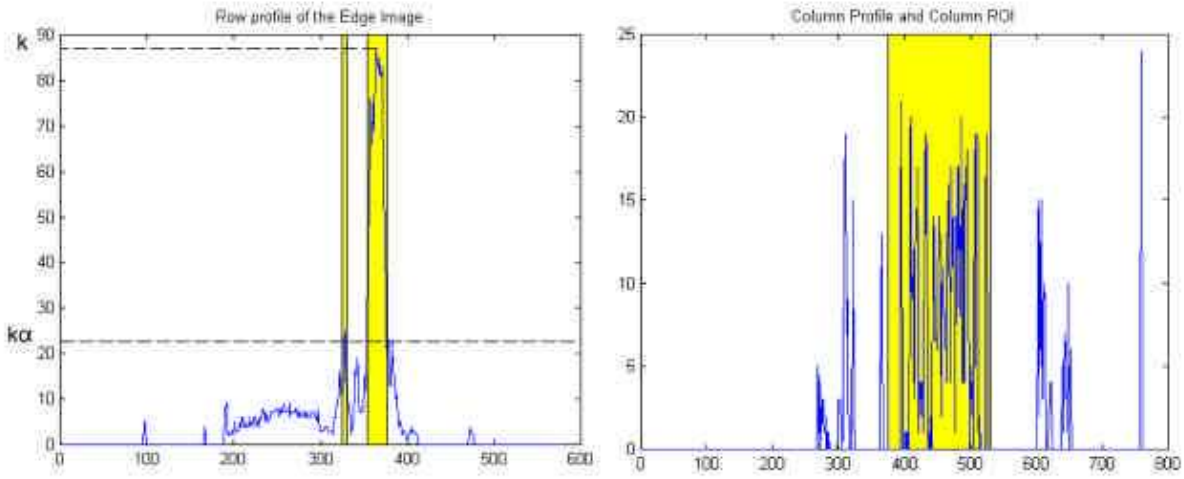

Figure 3: On the left-hand side we see the "row-profile" of the edge image. Each unit on the horizontal axis represents a row in the edge image. On the vertical axis the number of black pixels in that row is represented. The selected parts are the most likely positions for the rows of the license plate. On the right-hand side of the figure, we see the "column-profile" of the selected rows. The selected part corresponds to the most likely position of the license plate.

We created a so-called edge image corresponding to the greyscale photo. This was done by looking per row whether two consecutive pixels had a high difference in greylevel. If so, we replaced the first pixel of the two by a black pixel, otherwise by a white pixel. The result of this procedure on Figure 1 can be seen in Figure 2. Figure 2 is called the edge image corresponding to Figure 1. We see that there is a high number of edges on the position of the license plate. To select the region with a high number of edges, which is most likely to contain the license plate, we created a row profile. In a row profile we count for each row in the edge image the number of black dots and present this in a graph. The peaks of the graph indicate positions with many edges. In this way we can determine the most likely horizontal position for the license plate. We use the selected rows from the row-profile to create a truncated edge image (only the selected rows, nothing more). In this truncated image, we can make a column-profile: counting the number of black pixels for the columns. From the column-profile we can select the most likely vertical position of the license plate. In this way, we have found the position of the license plate.

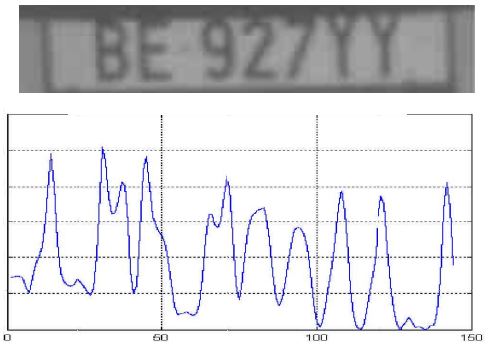

Figure 4: The column profile of the license plate area, created from a greyscale image.

The Segmentation Phase

The area that we found in the location phase is a part of the **original image**, and therefore a greyscale bitmap. We refer to it as the license plate area. We can look at the darkness of each column of this area. We expect a column that contains part of a character to be darker than a column that contains the background of the license plate[1]. So, again we create a column profile, but this time on a greyscale image instead of a black-and-white image. We can divide the license plate in characters by cutting it at the columns that are significantly higher than their surroundings. In this way, we obtain a number of small bitmaps, which contain one character each. This method is illustrated in Figure 4.

We obtain the character bitmaps if we cut on certain minima of this column profile (which minima we should select to cut is determined with an algorithm). The character bitmaps that we find are displayed in Figure 5.

[1] Note that we do not consider license plates with light characters on a dark background

Figure 5: The character bitmaps that are found as a result of the segmentation phase.

The Cleaning Phase

Now we have found the character bitmaps, we have to make sure that the computer can recognise the characters that they represent. To facilitate this, we first do some preprocessing; we clean the bitmaps. By this, we mean that we remove all dark pixels from the character bitmap that do not belong to the characters. Examples of such pixels are the black bars above and below the characters (see Figure 5), dirt spots, or nuts and bolts.

At the beginning of the cleaning phase, we convert the character bitmaps to black and white bitmaps by changing light pixels into white pixels and dark pixels into black pixels. From these black and white bitmaps we remove all rows and columns that contain only black pixels, since these are not part of a character, but of the black rectangle surrounding the license plate. Furthermore, we remove all small groups of black pixels. These are no characters either, but dirt spots, bolts, flies et cetera. The cleaning phase has as its output the greyscale input image, in which the found noise pixels are replaced by background pixels. The result of the cleaning phase applied to the character bitmaps in Figure 5 is displayed in Figure 6.

Figure 6: The cleaned character bitmaps. All dark pixels that are not part of the character are removed.

The Classification Phase

We use the greyscale character bitmaps obtained in the cleaning phase as input for the classification phase. In this phase, we determine which character can be found in the bitmap. We use a combination of methods for this, each method determining a certain feature of a character. The applied methods consider for example the number of peaks in the column profile of the character, the number of peaks in the row profile of the character, the number of end points of the character, or the number of junctions in the character. In Figure 7, the results of the classification method are visualised for a certain bitmap.

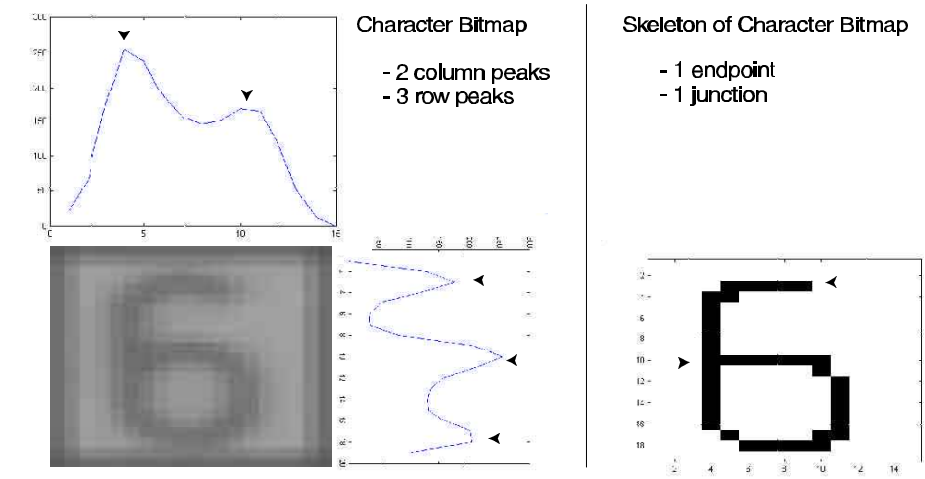

Figure 7: This character bitmap has 2 peaks in the column profile, 3 peaks in the row profile, 1 endpoint and 1 junction. Therefore, it has to be either a 9 or a 6.

As a result we see that the character in the bitmap has 2 column peaks, 3 row peaks, 1 endpoint and 1 junction. In a list of features we can look up that this holds for both a 6 and a 9. Therefore, we need another characteristic. This last characteristic has to distinguish between a 6 and a 9 only. The algorithm "knows" that the difference between a 6 and a 9 can be determine by looking at the row-position of the endpoint with respect to the row position of the junction. If the endpoint is higher than the junction, the character has to be a 6, otherwise it has to be a 9. All in all, the algorithm goes through the following steps:

1. The character bitmap might contain all characters. The algorithm starts with determining the number of peaks in the column profile and the number of peaks in the row profile.
2. The characters {5,6,8,9,B,G} all have 2 row peaks and 3 column peaks. The other characters cannot be contained in this bitmap. The second step of the classification algorithm is determined by the result of the first step. In this case, the algorithm determines the number of endpoints and junctions.
3. From the remaining characters {5,6,8,9,B,G}, the characters {6,9} are the only characters with 1 endpoint and 1 junction. The third step of the classification algorithm is dependent on the result of the second step. In this case, the algorithm determines the row position of the endpoint with respect to the junction.
4. From the remaining characters {6,9} is {6} the only character that has an endpoint that is higher than its junction. Conclusion: the character contained in this bitmap is a 6.

With this classification method, the problems of recognising characters that are very much alike is partly solved. The characters B and 8, 0 and D, and 2 and Z are still often not distinguishable, but the characters S and 5, V and Y, and C and G are.

Conclusion

The location and classification phase received in my final project the most attention, since the existing ALPS had the most problems there. Therefore, it was not surprising that these phases performed better than the segmentation and cleaning phase The overall result was satisfyable. It is now used, together with the already existing system in several automated character recognition systems.