18-794 Pattern Recognition Theory - License Plate Recognition
Carnegie Mellon University

Alexandre Alahi, Alok Menghrajani

April, 2003

Contents

1 Introduction

We designed and implemented an algorithm that takes raw car images and recognizes the license plate number. We worked with Swiss license plates.

We divided the problem in three parts: character recognition, character localization and license plate localization. Between the first and second part, we seperate the letters from the digits.

2 Specification

The license plates are black text on white background. There are three types : front (horizontal, 300mm x 80mm. The two letter state code is followed by a dot and then by the digits), back horizontal (500mm x 110mm. Starts with a colored Swiss icon, followed by the two letter state code, followed by a dot, followed by the digits and ends with a colored state icon) and back vertical (300mm x 160mm. This plate has two lines. On the top one, the colored Swiss icon is followed by the two letter state code, followed by the colored state icon. The digits are on the second line).

3 Telling apart digits from letters

Seperating the digits from the letters significantly reduces the complexity of classification. We tried to achieve this by finding the plate format. We found a solution that works in most cases.

3.1 Aspect ratio

The most intuitive method was to compare the aspect ratio with the specification. We used two thresholds, one of 2.8 and one of 4.1. The first threshold works very well and allows us to know if we have a horizontal or a vertical plate. The second threshold didn't give any results at all. Because more recent license plates have more digits. We ended up using only the first one.

3.2 Locating the dot

We tried to locate the dot that seperates the letters from the digits on the horizontal plates. Unfortunatly some license plates (very rare cases) don't have this dot. We also didn't want to rely on such a small quantity of pixels, which can in some cases be distorted, missing, etc...

3.3 Locating the icons

We tried several approaches to distinguish the Swiss and state icon with the pixels that are only black or white. We tried looking at every pixel's position in the rgb space. We tried to calculate the difference between the various corners. We also tried to work in the hsv [1] space. We weren't able to reach any stable results; the reason being the low resolution of the images and in some cases the color aberration of the camera.

[1] hue, saturation, value

4 Digit recognition

The goal of this part is to tell which digit is at a given location (knowing the bounding box).

4.1 Image preprocessing

In order to simplify the digit recognition algorithm, we preprocessed the initial image to get a similar image each time. Our processing order is the following: binarize the image, crop it and then make clusters.

We assume that the car is stopped (or that the camera's shutter speed is very fast) and we therefor are not trying to deblur the image against movement.

4.1.1 Binarizing the image

The image is first converted to gray scale. The gray value is simply the norm of the rgb vector. We then find a threshold that will seperate the pixels into black and white. We got the best results using the mean over all the pixels as the threshold. This allows us to binarize images taken at day time as well as images taken at night time.

4.1.2 Cropping the image

The image is then cropped by removing the white border around it. This makes all the images translation invariant.

4.1.3 Clustering the image

The pixels in the image are then grouped together (to form a smaller image). The idea of the algorithm is to average every pixel with it's neighbour. In our code, we used an extrapolation function (that takes a quadrilateral and morphs it into a rectangle) with a rectangle as input. The extrapolate function uses triangle mesh to calculate the values of the output rectangle based on the input points.

4.2 Digit recognition

We tried several methods for doing the digit recognition.

4.2.1 Histogram

We represented the spacial distribution of pixels by summing their values over the x and y axis. We then tried to classify the input image based on the shapes and values of the histograms.

This method was unable to distinguish the 5, 6, 8 and 9.

4.2.2 Concavity

Based on our own idea, we calculated the amount of pixels surrounded by black pixels, the amount this area represented and the center of gravity. We planned to use this method with the previous

histogram method in order to classify the 5, 6, 8 and 9. We got interesting results with this method, but it required high quality images.

4.2.3 Correlation

This method turned out to be very powerful. Since we have translation invariant images, we decided to use the correlation filters in the pixel domain (our code runs faster than correlation in the frequency domain). We used a training data of 7 images [2] (that gave us 38 images). We calculated 10 filters by averaging the images. We multiplied the image by each of these filters (pixel by pixel multiplication) and then summed over all the pixels and normalized the result. The classification is performed by simply taking the maximum correlation value.

4.3 Letter recognition

4.3.1 Correlation

Altough the principle for letter recognition is the same as digit recognition. The results we got were very mediocre since we are now dealing with a 19 class problem (some letters never appear in the state codes), so we had to find a different way of dealing with the same problem.

The training data to build the correlation filter was based on the entire second database (see section 8).

4.3.2 Valid state codes

We tried to calculate the correlation over the combined two letter code. For some odd reason we didn't get satisfying results. We continued with the same idea by creating a $mask$ matrix of valid state codes (19x19 matrix were the value at a given row,column can be either 0 or 1 (meaning that the state strcat(class{row}, class{column}) doesn't exist or exists respectivly). Since there are only 26 states, much of this $mask$ matrix is zeros.

Let a and b be the vectors of correlations between the first, respectivly the second, letter and the 19 classes. We take the maximum value of $a * b' * mask$.

We were very satisfied with the results this method gave us (section 9).

5 Character localization

The goal of this part is to find the digits on the license plate. The plate image can be rotated or even distorted due to perspective projection. Between this part and the plate localization, we need to transform the image using the same extrapolation function as before (see section 4.1.3). We morph the quadrilateral shape of the license plate into a flat rectangle. The resulting image implies that the digits are horizontally aligned.

[2] AG 101299, AI 30664, FR 90660, FR 59398, GE 52676, GE 442897 & NE 72783

5.1 Image preprocessing

In order to improve the results the image is preprocessed. The image is binarized using a threshold (matlab's graythresh function that uses 0tsu's method). The borders are suppressed (by scanning for rows that are more than 80% black).

5.2 Histogram

We calculate the black pixel distribution along the y (vertical) axis. If we are dealing with a vertical plate we detect the two overall peaks on the y axis, otherwise we detect the single peak. In order to find the height of the characters we use two methods. The first one consists of finding over the first quarter (from the peak) the first value under a threshold (based on the width of the plate). If this method fails, we try to find a local minima (again over the first quarter from the peak). When doing all this extraction, we make sure we are removing useless white spaces around the digits.

Once we have the character height, we create a histogram over the x axis. We divide the x axis by finding the white spaces (again we use a dynamic threshold depending on the height). Once we have done a first pass, we do a second pass to remove any eventual dot, and also correct errors based on the relative widths (if we have a large digit, we split it in two, as it is probably the combinaison of two digits). The maximum and minimum values for doing these error corrections are based on the one to greatest value.

6 Plate localization

We gave an attempt to locate the license plate on the image. This problem turned out difficult because we don't have any constrains on the image quality, car position, illumination conditions, etc...

6.1 Edge detection

We tried using edge detection methods. It turned out to be a very difficult task since we don't know the orientation of the plate, so we don't really know for what type of edges we are looking for.

6.2 Locating the white background

We find an approxmiate location using histograms. The exact position of the corners are found using a complex ladder algorithm. In short we move up until we are out of the white area. We then go right until we are again inside the white area, and up, and so on; the challenge is to stop exactly at the corners.

We filter the initial image using the imfill function. This function will removes the black pixels that cannot be reached from the image border. In the case of images taken at night time, this

removes the digits from the license plate.

7 Computational complexity

We estimated the computational complexity of our algorithm by using the profiler tool of matlab. Locating the plate takes 4 sec. Locating the characters takes 1 sec. Recognizing the digits takes 2 sec. The most complex operation is the extrapolation (morph).

8 Database

We seperated our images in three databases. We used the first two to train and test our code. The first database is a set of car images (27 images: 5 front, 11 back (horz), 7 back (vert), 2 bad - 16 night, 11 day). The second one is a set of license plates from every state (23 images). We kept the third database as our 'unknown' test set (40 images: 9 front, 11 back (horz), 15 back (vert), 5 bad - 21 night, 19 day).

9 Results

Here are the results we got while testing on our training set:

We have a 99.1% of recognition rate over the training database.

We have a 96.4% of recognition rate over the training database.

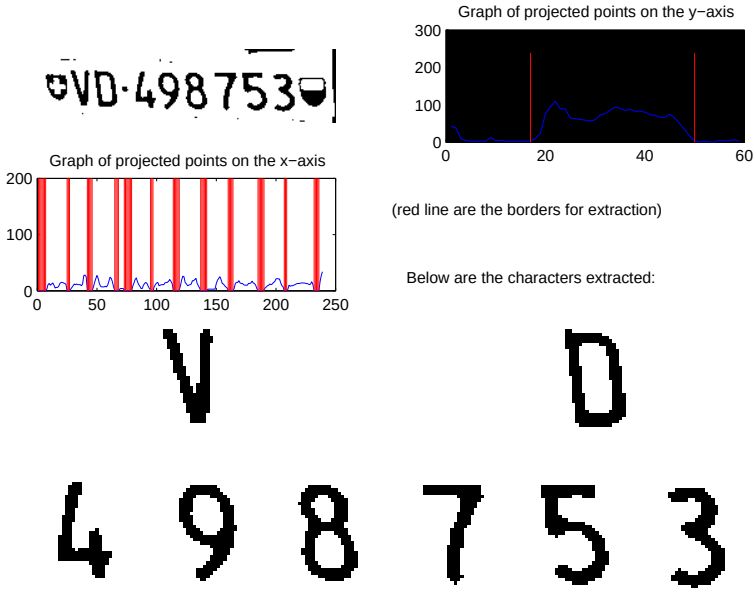
We have 100% of successful digit localization, but this result does not take in account our inability to process front license plates. The license plates were also selected manually.

We located 11 plates out of 27, considering some of the images were day time (which is by design the handicap of our algorithm) this is a reasonable result.

10 Running our code

We wrote a user interface in matlab. To run our code just set the path to our files and type ui. The user interface will let you manually perform the plate selection and character selection as well as run the code that does it automatically.

(Note: We will provide our code on a CD since the training data is large.)



Character localization.