18-794 Pattern Recognition Theory - Licence Plate Recognition

Alexandre Alahi, Alok Menghrajani

1 Introduction

We are going to design and implement an algorithm that takes raw car images and recognizes the licence plate number. We will be asuming that the car is stopped, but the environment can vary (day/night time, distance/angle, front/back view, etc...)

We will work with Swiss licence plates, the reason being the digits (4 to 6) are seperated from the two letter state code. At first we will work on digit recognition, and then later on, we will work on the state codes.

The front licence plate is 300mm x 80mm. There are two formats for the back: 500mm x 110mm and 300mm x 160mm. The plates are white, with the text in black and the font is always the same.

2 Database

We are building the image database on our own. We currently have over 50 pictures (we will take some more) with at least one sample of every state and between 25 to 100 samples of every digit.

3 Tools

We are working with Matlab and it's image processing toolkit. At first we will enter the plate and digit locations manually (we have already created an interface that permanently saves the information entered). If we have enough time, we will detect these positions automatically by searching for a white quadrilateral and checking intensity/pixel variances over the entire image.

4 Image Preprocessing

Since our database is raw images, we are going to do a lot of image preprocessing. Once we have the location of the plate, we will be working with greyscale or binary images, each pixel's value being in the range 0.0 (black) to 1.0 (white).

Among the ideas we have tried/thought:

- Enhance the contrast (this will allow us to seperate the digits from the background, but requires a dynamic threshold)
- Normalize the brightness (by taking the mean of the brightness and then multiplying every pixel by 0.5/mean)
- Smooth the edges (by removing small spures and filling small holes)
- Crop the digits so there are no white spaces arround them (centering)
- Clustering the image (reduce the number of pixels by doing an average over the neighbors)
- Stroke width normalization [1]

5 Feature Extraction

Our idea is to do a good job at the feature extraction level, in order to reduce the burden on the classifier.

Among the ideas we have tried/tought:

- Histogram of pixels [2] (we got interesting results by combining with pixel clustering)
- Quantity of pixels in concavity (our own idea)
- Edge detection (using predetermined patterns)
- Geometric moments (we haven't fully looked at this yet)
- Gabor filters and image correlations [3]

Using histogram of pixels along the x and y axis, we are able to differentiate between half the digits (the ones that caused trouble are 5, 6, 9, 8). We believe we will get the right results by enhancing our simple image preprocessing method and adding two more features from the list above.

6 Classification

A lot of the papers we found use neural networks. We even found some papers where no features were extracted and the neural network was fed in the images (probably preprocessed). The major problem with neural networks is the quantity of training data needed. We will therefore take an alternative approach.

Once we will have our features, we will reduce their size by using PCA. We will then use a nearest neighbor scheme to determine the digit.

7 Most interesting papers we found

[1] J. Hu, D. Yu, H. Yan: *Algorithm for Stroke Width Compensation of Handwritten Characters*, Electronics Letters, 21st Nov 1996, Vol.32, No.24.

[2] O.D. Trier, A.K. Jain, T. Taxt: *Feature Extraction Methods for Character Recognition - A Survey*, 1995.

[3] T. Wakahara, Y. Kimura: *Affine-Invariant Gray-Scale Character Recognition Using GAT Correlation*, IEEE 2000.

[4] G. Srikantan, D. Lee, J.T. Favata: *Comparison of Normalization Methods for Character Recognition*, IEEE 1995.

Example of a raw image and the different formats for the plates.