3-2015

An End-to-End License Plate Localization and Recognition System

Siyu Zhu

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation


This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
An End-to-End License Plate Localization and Recognition System

by

Siyu Zhu

A Thesis Submitted in Partial Fulfillment
of the Requirements for the
Degree of
Master of Science
in
Electrical Engineering

Approved by:

Dr. Sohail Dianat, Thesis Advisor

Dr. Lalit K. Mestha, Committee Member

Dr. Eli Saber, Committee Member

Dr. Sohail Dianat, Department Head

Department of Electrical Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
March 2015
Title:

Electrical Engineering
M. S. Thesis:
An End-to-End License Plate Localization and Recognition System

I, Siyu Zhu, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

Siyu Zhu

Date
Acknowledgments

I am grateful for Dr. Sohail Dianat’s advise on algorithms and system design. Thanks also go to Dr. Lalit K. Mestha for his advise on heuristic rules selection and system performance evaluation.
Abstract

An End-to-End License Plate Localization and Recognition System

Siyu Zhu

Supervising Professor: Dr. Sohail Dianat

An end-to-end license plate recognition (LPR) system is proposed. It is composed of pre-processing, detection, segmentation and character recognition to find and recognize plates from camera based still images. The system utilizes connected component (CC) properties to quickly extract the license plate region. A novel two-stage CC filtering is utilized to address both shape and spatial relationship information to produce high precision and recall values for detection. Floating peak and valleys (FPV) of projection profiles are used to cut the license plates into individual characters. A turning function based method is proposed to recognize each character quickly and accurately. It is further accelerated using curvature histogram based support vector machine (SVM). The INFTY dataset is used to train the recognition system. And MediaLab license plate dataset is used for testing. The proposed system achieved 89.45% F-measure for detection and 87.33% accuracy for overall recognition rate which is comparable to current state-of-the-art systems.
## Contents

Acknowledgments ........................................ iii

Abstract ........................................ iv

1 Introduction ........................................ 1

2 Previous Work ...................................... 4
   2.1 Localization .................................. 5
   2.2 Segmentation .................................. 6
   2.3 Recognition .................................. 7

3 License Plate Detection ............................ 9
   3.1 Preprocessing .................................. 9
      3.1.1 Bilateral Filter ......................... 9
   3.2 Thresholding .................................. 12
   3.3 Connected Components Properties Extraction .... 13
   3.4 Spatial Relationship Check .................. 14

4 Character Extraction and Segmentation .......... 17

5 Character Recognition ............................ 21
   5.1 Feature Extraction ............................ 21
   5.2 Classification ................................ 22
      5.2.1 Alignment ............................ 22
      5.2.2 Curvature Histogram .................. 26
6 Experiment Result and Analysis ........................................ 28
  6.1 Dataset .............................................................. 28
    6.1.1 MediaLab Dataset ........................................... 28
    6.1.2 \textit{INFTY} Dataset ....................................... 29
  6.2 Ground Truth and Evaluation Metric Design ...................... 30
  6.3 Detection Performance Evaluation ................................ 31
  6.4 Character Segmentation and Recognition Evaluation .......... 33
  6.5 Performance Comparison .......................................... 37

7 Conclusion ............................................................. 39

Bibliography ............................................................. 41
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Properties checked for Object Detection.</td>
<td>13</td>
</tr>
<tr>
<td>3.2</td>
<td>Properties checked for Spatial Relationship.</td>
<td>13</td>
</tr>
<tr>
<td>6.1</td>
<td>Number of samples in different categories of MediaLab license plate dataset.</td>
<td>29</td>
</tr>
<tr>
<td>6.2</td>
<td>Precision, Recall and F-metric of the license plate detection system.</td>
<td>32</td>
</tr>
<tr>
<td>6.3</td>
<td>Accuracy for segmentation, recognition and overall system for different categories of images.</td>
<td>36</td>
</tr>
<tr>
<td>6.4</td>
<td>Accuracy for different categories of images using turning function (TF) and curvature histogram (CH) based SVM.</td>
<td>37</td>
</tr>
<tr>
<td>6.5</td>
<td>Existing license plate recognition systems, which used different testing datasets and could not be compared directly with proposed system.</td>
<td>38</td>
</tr>
<tr>
<td>6.6</td>
<td>Proposed system performance comparing to Anagnostopoulos’ system using NTUA MediaLab dataset [2].</td>
<td>38</td>
</tr>
</tbody>
</table>
6.7 Proposed system computation time comparing with Anagnostopoulos’ system.
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Typical Automatic Plate Recognition System.</td>
<td>2</td>
</tr>
<tr>
<td>3.1</td>
<td>Examples of images from MediaLab dataset.</td>
<td>10</td>
</tr>
<tr>
<td>3.2</td>
<td>Proposed License Plate Recognition System.</td>
<td>10</td>
</tr>
<tr>
<td>3.3</td>
<td>The original image and the image after bilateral filtering.</td>
<td>12</td>
</tr>
<tr>
<td>3.4</td>
<td>The example of the license plate detection system. First, the image is binarized using adaptive threshold (b). Then objects are extracted as CCs (c). The CCs are filtered using property check (d). Remaining CCs go through morphological transform, including dilation (f) and erosion (g). After that, spatial relationship check is performed as another filter. The final output is the correct location of the license plate (i).</td>
<td>15</td>
</tr>
<tr>
<td>4.1</td>
<td>Example of Image Segmentation using Projection Profile. (a) Region of Interest; (b) Vertical Projection Profile; (c) Peak Detected Without Elevation Difference Check; (d) Peak Detected with Elevation Difference Check.</td>
<td>18</td>
</tr>
<tr>
<td>4.2</td>
<td>Example of peak extraction using FPV technique. (a) Initial Peak; (b) Valley candidate; (c) Valley found with higher peak; (d) Move peak to the higher point.</td>
<td>19</td>
</tr>
<tr>
<td>5.1</td>
<td>Turning function generation of training and testing sample.</td>
<td>23</td>
</tr>
</tbody>
</table>
5.2 Horizontal and Vertical shift of turning function due to starting point movement and rotation. (a) Horizontal shift of turning function by shifting starting point; (b) Vertical shift of turning function by rotation. .......................... 24

5.3 Turning function of training sample and testing sample. (a) Turning function training sample; (b) turning function of testing sample; (c) Training, Testing sample alignment. . . . 25

6.1 Graphic User Interface (GUI) of the license plate detection system. (a) The image loading interface; (b) The raw image with detected license plate region; (c) The output text file containing bounding box coordinates and plate character labels. ................................. 31

6.2 Example output of license plate detection system. Some error detection examples are shown, where license plate is missed or detection region is larger than true region of license plate. ................................. 32

6.3 Example output of license plate recognition system. The detected region (as shown in second row) is firstly binarized using adaptive thresholding, and projection profile and FPV is used to segment binary image (third row) into individual characters. The output labels are shown in captions below image pairs. ................................. 34
6.4 Hard examples of license plate detection and recognition. (a) Successful detection but failed segmentation/recognition, due to blurring; (b) successful detection and segmentation, but wrong recognition, due to blurring and break character shapes; (c) Missing character due to wrong segmentation of touched characters; (d) Wrong recognition due to wrong segmentation of very small characters; (e) Missing character, due to touching noisy objects; (f) Character is wrongly recognized due to high skewing; (g) Wrong recognition due to high skewing and low contrast; (h) Successful recognition with strong luminance inhomogeneity.
Chapter 1

Introduction

In this paper, an end-to-end system for license plate recognition (LPR) is proposed, which means the input of the system is raw camera based images and output is character labels. It consists of three components: license plate localization, character segmentation and character recognition. Pre-processing is to solve illumination and color problems, while post-processing is used to correct view angles and compensate skewness of the images.

The LPR technology is widely implemented in security and traffic installations. A camera is used to capture the image of plates in the front of or rear of vehicles. The license plate number is recognized and stored in the information system for automatic law enforcement, traffic control and facility management. Some system requires only conventional cameras to collect data, while some others need assistance of special equipment, e.g. infrared camera, special illumination, motion sensor. In this paper, we utilize still image captured from conventional digital cameras. Flashlight is used during night. A typical automatic license plate recognition system architecture is as shown in Figure 1.1. It consists of image requisition, processing (LPR) and information retrieval components.

Among current LPR systems, the simplest example is that of European license plates. It contains black numerical and English characters on white background. A more difficult example is United States license plate. This kind of plates contains figures, state names, different fonts and colors. The variations of colors and shapes bring additional challenges. Customized license plates make the problem even more complex. A third category is like Chinese or Japanese license plate. These kinds of license plates have similar
layouts as of the first category, which make it possible to detect using similar algorithm. However, Chinese and Japanese contains larger vocabularies and complicated fonts, which are very difficult for recognition, especially when resolution is low.

Text detection systems are studied extensively during recent decades. They usually consist of several steps to locate, extract, segment, and recognize text from complicated image scenarios, like the natural scenes. Similar architecture consisting of localization, segmentation and recognition components could also be used for license plate detection. Preprocessing is usually used to enhance contrast, calibrate colors and detect edges making the text more detectable for the next stages. Image binarization is usually used to convert color image or grayscale image into binary valued image, with only 0 and 1 for each pixel indicating foreground and background. Then a detection system is used to locate the text position within the image by checking the properties of the connected components (CC)s and their relationships. After the position is detected, a bounding box of the detected license plate is marked and content within the bounding box is cropped for
further processing. Some de-skewing system might be needed for images taken from various aspect angles. A second binarization maybe needed to extract text from the cropped bounding box region. Then a segmentation algorithm is used to divide the text within the region of interest into individual characters. Each single character is then put into an Optical Character Recognition (OCR) system. The label of each character is then assigned.

Unlike generic text detection systems, license plate detection system does not care about texts other than those on license plates. In some cases, this makes the task easier, because some hard-to-see and highly skewed texts in the background do not need to be detected or recognized. On the other hand, text looks like license plate will generate false positives.

The proposed system focuses on European automobile licenses using MediaLab dataset [2] for training and testing. Because the dataset is publicly available, the performance could be fairly compared with previous systems. As European license plates are easier to process, it could be used as a basic module of the LPR system. Extra processing components could be added according to specific types of targets, like Chinese or US plates. However, we also have addressed the problems that might arise in other types of license plates, in order to make the system more robust and generalized.

The remaining sections are arranged as follows: Chapter 2 surveys the previous work on LPR; Chapter 3 introduces the proposed algorithms and methods; Chapter 4 details the building of the experimental system; Chapter 5 presents the recognition results and analysis; Chapter 6 is the conclusion.
Chapter 2

Previous Work

Recently, many works have been done towards different kinds of license plates, like Chinese [40, 23, 37, 25, 13, 36, 45, 24] and Arabic [26, 6, 18, 1]. A survey done by Caner et. al. [8] covers many prototype models for LPR. Most systems use still images captured by conventional camera as input. However, some systems utilize additional information to achieve higher accuracy. Davies et al. [11] proposed license plate reading system (3MRIA-300) for toll violation enforcement. This system used two observations (cameras) to read the license plate and to correct each other to achieve higher accuracy. Arth et al. [5] used Kalman filtering method to predict the future position of the license plate in frame sequences captured by video cameras. In this paper, we focus on LPR systems using still images as input. Image processing aspect of the system is discussed. Because no additional equipment, e.g. infrared-camera, is needed, the installation cost is minimized. Still-image based system could capture higher resolution images for processing and require smaller data storage space, while video based systems can continuously record traffic conditions. Each of them has certain advantages, and should be used according to specific purposes.

Image based LPR systems are composed by three major processing components: detector, segmenter and recognizer. Detector localizes the position of the license plate from raw images, allowing further processes only be performed in the region of interest. Segmenter divides the region of interest into individual characters. Therefore, the recognizer could classify each one into a text label.
2.1 Localization

Many different features are utilized to find the location of the license plates, e. g. shape, color, orientation and frequency.

Colors and shapes of the objects are among the most popular features that are utilized [28, 27, 33, 22]. Some systems [20] change the color space to find more robust representations.

Projection profiles are used in Dai’s work [41] and Wu’s work[39] to find the locations. Edge detection is often used to help locate license plates. Systems detect vertical edges to find two parallel edges on each side of the license plate, or to find characters [19, 32, 1]. In Hsieh’s system [16], the shape of the edge is firstly checked to find the candidate of the license plate. Then location is found using density, size and aspect ratio etc. A recovery system is proposed after detection. The region of interest will slightly grow in size, until a desired number of characters are identified or the region area meets the maximum. Bai et al.[15] combines edge detection and CC filtering method to detect license plate in images. In Faradji’s [12] system, compact factor is introduced, therefore, objects like fences will not be confused with license plates. Similarly, in Pan’s paper [29], vertical edges and morphological transformation are used to find the targets.

Crossing count (or cross-section) is also used in some system to find the location [31, 38]. Crossing count is the number of crosses between edges and pre-fixed base lines.

Besides color and shape information, frequency is another useful feature to find the position of the plates. The vertical frequency analysis is used to detect the location of the license plate in Huang’s work [17]. The areas with high density of mono-oriented gradient are selected. In Koval’s work [21], first the image is de-blurred using a multi-resolution transform. The mean contrast frequency is calculated for each position of template. Then the de-blurred image is created by thresholding the template-matching map. The geometrical information of object is used to find the location.

Orientation information is utilized in Anagnostopoulos’s [2] work. Log Gabor features which address the local orientation and frequency is used to find the region of the license plate.
Machine learning algorithms were utilized, combining different features to increase accuracy while reducing the execution time. Neural networks are used in many LPR systems [22, 35, 9]. Discrete Time Cellular Neural Network (DTCNN) is used by Ter’s [35] and Chang’s [9] system, due to its good performance on image based problems. In Mei’s work [43], the car is firstly detected using gradient of color variation. Given that the background of the images are all homogeneous, a high gradient only appears when a car is inside the image. Then license plate is identified using two TDCNN classifiers for vertical and horizontal cross-segments respectively. Finally, detection results from the two classifiers are combined to find the location of the license plate.

Fuzzy algorithms are also implemented in detection. In Comelli’s work [10], fuzzy C-means is used to detect the location of the license plate. Chang et al. [9] used fuzzy maps to detect the locations of the license plates. The edge, hue, intensity and saturation fuzzy maps are used to build a fuzzy cost function for license plate location in their algorithm.

Meanwhile, Viola&Jones-like cascade AdaBoost detector is used to find the location of the license plate by Arth et al. [5]. The Real-AdaBoost algorithm is used to gain improved accuracy. A sliding window method called Sliding Concentric Window (SCW) is used to detect the location of the license plate in Anagnostopoulos et al.’s work [3]. It used two sliding windows, sharing the same central point pixel, but with different window sizes. The ratio of the statistical measurements inside a couple of windows is used to classify the current pixel into foreground or background.

### 2.2 Segmentation

After localization, the region of interest is segmented so that each individual character is separated. For segmentation, projection profiles are popular among LPR systems. [22, 22, 41, 44, 32, 1, 2] Zhang’s system [44] used Hough transform to aid the horizontal profile projection segmentation to achieve better result. In addition, the skew correction system for license plate is proposed to solve the tilting problem by Pan[30]. Changing of camera viewing angles usually causes the tilting. Horizontal skewing is
compensated using direction angle histogram. The image is divided into a set of non-overlapping blocks. And orientations of blocks are computed. The image is then transformed using the peak of the histogram of orientations (HOG). The vertical skewing is compensated using the single character projection profile. The characters are separated into upper and lower parts. Their projection profiles are aligned in order to align the license plate images.

2.3 Recognition

After localization and segmentation, the characters are recognized to generate text labels. Template matching is popular among LPR systems to recognize individual characters \([22, 28, 19, 27, 41, 32, 17, 39, 18, 29]\). The distance measure of the templates is usually Euclidean or correlation. However, other distances could be used to achieve more generalized comparison, like Hausdorff distance \([18]\). To increase speed, neural networks \([10, 35, 33, 31, 21, 9, 3]\) and SVMs \([43, 5]\) are trained for classification. In Sirithinaphong’s \([33]\) system, license plates are located using the regulations in terms of shapes, colors and aspect ratios. The detected license plate is segmented using projection techniques. Each character is recognized using a 4 layer neural network. Chang et al. \([9]\) proposed to use Kohonen’s neural model to recognize characters. The ambiguous characters are further distinguished using additional minor comparison. In Anagnostopoulos’s work \([3]\), the characters are recognized using probabilistic neural networks (PNN).

After recognition, the accuracy could be further increase by using lexicon rules and language model. For example in Comelli’s work \([10]\), as the final step of the system, some syntactical rules are applied to further increase the accuracy of recognition.

Based on previous work, we propose a detection system that utilize both color and shape information to find the license plate positions. Furthermore, the spacial relationship between characters is also analyzed to increase the localization accuracy. A projection profile based segmentation method is proposed to cut the region of interest into characters. Elevation check and
floating peaks and valleys (FPV) algorithm is proposed to find meaningful cutting point of the profiles. A template matching method based on turning function is proposed to recognize characters. The turning function method will convert 2-D images into 1-D functions, while preserving the shape information. It could also compensate the distortion, including scaling, skewing and rotation of the characters. We further accelerate the system by using a SVM classifier and histogram of curvature functions.

Comparing to previous systems, the proposed LPR system combines color, shape and spatial relationships to locate the license plate more accurately. New algorithms are proposed to cut projection profiles quickly to segment region of interest. And new features are proposed to recognize text labels with high accuracy and fast execution speed.

Our algorithm is tested on a publicly available still image dataset, the NTUA MediaLab dataset [2]. Although testing images contains many noises, transformations, blurring, luminance inhomogeneity and other types of difficulties, accurate performance and fast execution are achieved. The system could perform localization, segmentation and recognition solely based on wild scene images without other information, e.g. motion sensor data, infrared camera data. This technique could be combined with other techniques to get higher accuracy, or to be used in cases where images are from different sources and significant noise and deformation exists.
Chapter 3

License Plate Detection

We designed an end-to-end license plate detection and recognition system. The system combines license plate detection and character recognition to output an informative result with license plate coordinates and character labels. The detection system includes preprocessing, thresholding, CC grouping, target object extraction. After a candidate region is detected, characters in license plate are segmented. Then each individual character is recognized using their contour information contained in the feature called turning function. We tested our system on MediaLab dataset [2]. Some examples of the images from this dataset are shown in Figure 3.1. Figure 3.2 shows the proposed system that consists of preprocessing, binarization, localization, segmentation and recognition. The inputs are RGB color images, while the outputs are bounding-box locations of the license plate and the associated recognition results.

3.1 Preprocessing

The noise in the images is a big obstacle for extracting accurate text regions. It is almost impossible to avoid for conventional camera based images. However, noise removal approaches have been improved in the recent years. An effective algorithm used in our system is bilateral filtering.

3.1.1 Bilateral Filter

The conventional filtering methods for noise removal have a common problem: the filter blurs true edges when removing noise points. To counter this
Figure 3.1: Examples of images from MediaLab dataset.

Figure 3.2: Proposed License Plate Recognition System.
problem, the bilateral filter is designed to penalize both distance and pixel value difference when averaging over a sliding window. Therefore, the pixels with values similar to the current center will have larger weights, while pixels with values different from current center will have smaller weights. On the other hand, the weights also vary based on the geometric distances between the pixels and the center. The pixels farther away will have smaller weights, while closer pixels will have larger weights. Usually two discrete Gaussian distributions will be used to model the varying weights of the pixels. Hence the name bilateral, as shown in Equation 3.1 and 3.2.

\[
\begin{align*}
  w(i, j, k, l) &= e^{-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2} - \frac{||I_{i,j}-I_{k,l}||^2}{2\sigma_r^2}} \\
  I_D(i, j) &= \frac{\sum_{(k,l)} I(k, l) w(i, j, k, l)}{\sum_{(k,l)} w(i, j, k, l)}
\end{align*}
\]

(3.1) (3.2)

where \(w(i, j, k, l)\) is the weight for pixel at position \(k\) and \(l\), \(i\) and \(j\) is the current center position. \(I_{i,j}\) and \(I_{k,l}\) is the intensity of center and neighboring pixels respectively. \(\sigma_d\) is the variance of distance kernel, and \(\sigma_r\) is the variance of intensity kernel.

The bilateral filter has been shown to work well in the presence of salt and pepper noise for camera based images, however, it is more computational expensive to use than Gaussian filters. This is because Gaussian filters only consider the pixel positions, so a convolution could be used to quickly produce result in one shot. On the other hand, the bilateral filter needs the information for both pixel position and pixel value, which doesn’t allow it be computed in one pass. The bilateral filter can only be processed in each sliding window for each center pixels, and the Gaussian average is computed for position and pixel value difference for each window and throughout the entire image. Therefore, bilateral filter is slow comparing to the basic Gaussian filter, but could produce much better results in terms of preserving true edges in images.
Figure 3.3: The original image and the image after bilateral filtering

3.2 Thresholding

After noise removing, we binarize the pixel values to 0s and 1s. A locally adaptive threshold is used to convert the real valued pixels into binary values with 0s and 1s. Global histogram based thresholding methods, including Otsu’s method do not perform well in our experiment, due to variation of illuminations. For this reason, we use a simple but powerful adaptive thresholding method [14]. The threshold value is based on the average of the current window. However, in order to decrease noise in homogeneous regions, the threshold $t$ is adjusted by a small constant offset $c$. $c$ is selected small enough, so that the binarization will produce clear boundaries along true edges. For pixels in the homogeneous region, where the variance of intensity is small, instead of getting on (1) and off (0) randomly, they are more turned to be classified as single region. Equation 3.3 shows how to compute a threshold for local image patch:

$$t = \frac{\sum_{i,j \in \Omega} I_{i,j}}{n} - c$$  \hspace{1cm} (3.3)

where $t$ is the local threshold, $I_{i,j}$ is the intensity value at position $i, j$, $\Omega$ is window centered at current pixel, $n$ is number of pixels in the neighborhood,
and \( c \) is the constant Mean-offset value that we pick. Constant \( c \) is decided using a grid search, where \( c = 10 \) is used in our system.

### 3.3 Connected Components Properties Extraction

After binarization, connected pixels are grouped into connected component (CC). Since European license plates contain dark texts in bright background, we assume the 0 valued pixels to be the candidates for texts. However, for other types of license plates, we could extract regions for both 0s and 1s, then find out which is more likely to be true text region.

<table>
<thead>
<tr>
<th>Property</th>
<th>Definition</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centroid</td>
<td>Center pixel coordinates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Width</td>
<td>-</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>Height</td>
<td>-</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>Aspect Ratio</td>
<td>width / height</td>
<td>1/8</td>
<td>2</td>
</tr>
<tr>
<td>Perimeter</td>
<td>-</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>Area</td>
<td>Total No. of Pixels</td>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>Bounding-box Area</td>
<td>Area / Bounding-box Area</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convex Area</td>
<td>Smallest convex shape that cover object</td>
<td>30</td>
<td>200</td>
</tr>
<tr>
<td>Filled Area</td>
<td>Area after filling the holes</td>
<td>30</td>
<td>140</td>
</tr>
<tr>
<td>Solidity</td>
<td>Area / Convex Area</td>
<td>0.3</td>
<td>1</td>
</tr>
<tr>
<td>Extent</td>
<td>Area / Bounding-box Area</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>Major Axis Length</td>
<td>Variance along 1st principle component</td>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>Minor Axis Length</td>
<td>Variance along 2nd principle component</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>Eccentricity</td>
<td>Major Axis Length / Minor Axis Length</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>Holes</td>
<td>-</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Definition</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>-</td>
<td>100</td>
<td>400</td>
</tr>
<tr>
<td>Height</td>
<td>-</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>Aspect Ratio</td>
<td>width / height</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>Perimeter</td>
<td>-</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>Area</td>
<td>Total No. of Pixels</td>
<td>100</td>
<td>800</td>
</tr>
<tr>
<td>Bounding-box Area</td>
<td>Area / Bounding-box Area</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extent</td>
<td>Area / Bounding-box Area</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>Eccentricity</td>
<td>Major Axis Length / Minor Axis Length</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>Orientation</td>
<td>Angle of 1st principle axis</td>
<td>-15</td>
<td>+15</td>
</tr>
</tbody>
</table>

The properties of each CC are computed and extracted. CCs that failed
property test are eliminated. The remaining CCs, which passed all property tests, are candidates. It is hard to achieve both high recall and precision in one step. Therefore, the thresholds are tuned to get moderate false alarm rate while keeping the recall sufficiently high. The extracted properties are: centroid coordinates, perimeter, area, bounding box area, extent, major axis length, minor axis length, eccentricity, number of holes, convex area, filled area, solidity etc., as seen in Table 3.1.

For example, centroids are the center pixel coordinates. They are also the average x, y coordinates of pixels within each object. A threshold is used that centroid of license plate objects are 5 pixels away from image boundaries. Thresholds used for other properties are shown in Table 3.1. After heuristic rule thresholding, candidate regions are selected. Criterions are selected so that high false alarm rate is produced to keep high recall. To eliminate unwanted objects, a second stage filtering is designed to check spatial relationships.

### 3.4 Spatial Relationship Check

Because of the nature of design of license plates, characters are usually arranged in a line with inter-character distance less than a typical character width. The entire license plate heights are a few pixels larger than the typical character height. Utilizing this spatial relationship between characters, one can eliminate noise objects and get improved precision for detection.

Many systems have been proposed to detect text utilizing spatial relationships. However, most of them compute the distance between each pair of characters to form matrix of character relationships. For example, for a 6-characters license plate, a 6 by 6 matrix is established and 36 relationships are examined to classify the current structure. The noise produces many false positive objects inside an image, thus a large number of possible combinations require to be considered, leading to the need of building a classifier to extract the candidate region from all possible combinations. Therefore, this is not computationally feasible for low precision candidates.

We propose the use of morphological transformation to extract candidate license plate regions. This is a very efficient and straightforward method,
Figure 3.4: The example of the license plate detection system. First, the image is binarized using adaptive threshold (b). Then objects are extracted as CCs (c). The CCs are filtered using property check (d). Remaining CCs go through morphological transform, including dilation (f) and erosion (g). After that, spatial relationship check is performed as another filter. The final output is the correct location of the license plate (i).

which could utilize the spatial relationship between characters and improve detection precision significantly. The morphological operation is similar to Faradji’s system [12], but the operation is performed directly on CCs instead of on edge maps.

As characters in license plates are aligned in lines while surrounded by rectangle bounding box. We could assume that all characters inside license plate will group into a rectangle-like shape if we dilate those CCs by some
structuring element (SE) of a given size. Here we pick the structuring element such that CCs are dilated horizontally. The structuring element is designed to have a horizontal extent about half of the inter-character space, to ensure characters merging into one single large CC with a rectangle-like shape. After grid search, the dilation kernel is fixed to 10 pixels in width and 1 in height. After dilation, morphological erosion using the same kernel is performed. Obviously, the dilation will not only connect the true characters to form a large rectangle, but also extent the rectangle width at left and right boundaries. Therefore, erosion corrects the rectangle width to the actual width of the license plate. After dilation and erosion, another CC extraction is performed to extract license plate from the remaining candidates. We check the properties of morphologically transformed CCs and filter objects using similar heuristic rules. But this time, the property check is to find rectangular objects, not character-like objects. The width, height, aspect ratio, area, extent and orientations are used to classify the target regions. All properties that checked for spatial-relationship are listed in Table 3.2.

After CC filtering based spacial relationship checking, our system extract most of the license plate correctly, the false alarm has been reduced significantly. Meanwhile, some false positives still exist, which could be eliminated in recognition process. An example of license plate detection using our system is shown in Figure 3.4.
Chapter 4

Character Extraction and Segmentation

Once license plates are located, individual characters from the detected region have to be segmented. In our algorithm, projection profile is used to remove noise and separate characters. The projection profile counts number of foreground pixels along perpendicular lines towards a specified direction and draws a histogram, which reflects the number of pixels in each line.

Profiles are projected from two orthogonal directions, vertical and horizontal. The horizontal profiles are used to remove noise above or beneath the license plates. The vertical profiles are used to segment license plates into individual characters. The horizontal histogram shows two peaks at the head and tail of the distribution corresponding to the two plate edges at the top and the bottom. Therefore by detecting the two valleys near these two peaks, one could easily extract text region from the license plate region.

For vertical projections, the location of the peaks and valleys are needed to segment individual characters. Many segmentation methods have been proposed. Bissau et al. [7] proposed using Radon transform to segment texts, Yoo et al. [42] proposed to find cutting points in projection profiles recursively. We propose a algorithm to cut projection profiles quickly and accurately in time $O(n)$ ($n$ is the number of points in a profile), which is called Floating Peak and Valley (FPV) method. There is noise inside the projection profile, as we can see in Figure 4.1. The shapes of characters and objects are complicated. If every inflection point is used, too many peaks and valleys are produced, as shown in Figure 4.1c. Therefore, elevation difference check is introduced to extract only meaningful peaks and valleys. if the difference is greater than a threshold $\delta$, a valley or a peak point is marked. Thus, a peak or valley point is recorded only when there is
Figure 4.1: Example of Image Segmentation using Projection Profile. (a) Region of Interest; (b) Vertical Projection Profile; (c) Peak Detected Without Elevation Difference Check; (d) Peak Detected with Elevation Difference Check.

A search for peaks and valleys is performed inside a project profile from left to right. A method called floating peaks and valleys (FPV) is introduced, as in Figure 4.2. When searching for a valley point, the previous peak point is floating with cursor as in Figure 4.2a and 4.2b. If a point is found with a higher elevation, as shown in Figure 4.2c, we move the previous peak point to the new location instantly, as in Figure 4.2d. It ensures that peak point is always the highest point between two valleys. The floating peak will stop moving, once a new valley point is added. And the peak point is fixed to the current location. Then, alternatively, a peak point is searched for with a floating valley point. By altering floating peaks and floating valleys, we have make sure that all peaks are maximum points between two valleys and valleys are the minimum points between two peaks. Together, with
Figure 4.2: Example of peak extraction using FPV technique. (a) Initial Peak; (b) Valley candidate; (c) Valley found with higher peak; (d) Move peak to the higher point.

elevation difference check, meaningful peaks and valleys are extracted from the histogram without false detections along zigzag curves. The pseudocode of searching for peaks and valleys using elevation check and FPV technique is shown below. Projection profiles are given in \((x_i, y_i)\) pairs, where \(x_i\) is the horizontal coordinates, and \(y_i\) is the profile value at corresponding position. \(\delta\) is the elevation threshold. The difference between peak and valley must be greater than \(\delta\) to mark that peak/valley point.

The valley points will be used to segment characters. There, each region contains a single character. However, some regions might contain multiple characters where two characters are touching. The connecting characters will be further processed using recognition results.
Procedure 1 searchForValleys()
   for $x_i$ moves along projection profile $(x_1, y_1), (x_2, y_2), ..., (x_i, y_i), ..., (x_n, y_n)$ do
      highest peak altitude $p_{\text{max}} = 0$
      if Current point $x_i$ is a peak point and $y_i > p_{\text{max}}$ then
         Move peak point $p$ to $x_i$
         $p_{\text{max}} = y_i$
      end
      if Current point $x_i$ is a valley point and $|p_{\text{max}} - y_i| > \delta$ then
         add peak point $p$ as a fixed peak
         searchForPeaks()
      end
   end

Procedure 2 searchForPeaks()
   for $x_i$ moves along projection profile $(x_1, y_1), (x_2, y_2), ..., (x_i, y_i), ..., (x_n, y_n)$ do
      lowest valley altitude $v_{\text{min}} = \text{Inf}$
      if Current point $x_i$ is a valley point and $y_i < v_{\text{min}}$ then
         Move valley point $v$ to $x_i$
         $v_{\text{min}} = y_i$
      end
      if Current point $x_i$ is a peak point and $|v_{\text{min}} - y_i| > \delta$ then
         add valley point $v$ as a fixed valley
         searchForValleys()
      end
   end

Procedure 3 isPeak()
   if $y_i > y_{i+1}$ and $y_i > y_{i-1}$ then
      return True
   else
      return False

Procedure 4 isValley()
   if $y_i < y_{i+1}$ and $y_i < y_{i-1}$ then
      return True
   else
      return False
Chapter 5
Character Recognition

One way in which characters can be identified is by their shape. In this paper, character recognition is performed using turning functions. It is originally proposed by Arkin et al. [4] to compare polygonal shapes. We have used in the past for Math Symbol Recognition [47] with great accuracies. Turning function transforms a 2-dimensional matrix into a 1-dimensional vector while preserving shape information. Turning functions are generated from \( x, y \) coordinates of boundary points of characters. The alignment algorithm is implemented to compensate vertical and horizontal shift of turning functions. Finally, a nearest neighbor classifier is used to identify characters according to correlations. Based on turning functions, curvature histograms are proposed. It can accelerate classification speed significantly as no alignment is required. The accuracy and speed of turning function and curvature histogram are compared in our experiment, and shown in Section 6.4.

5.1 Feature Extraction

The \( x, y \) coordinates of the boundary points of individual characters are extracted in a counterclockwise direction, as shown in Figure 5.1. An alignment procedure as shown in Section 5.2.1 is introduced to find starting points. A sliding window average is used to smooth turning function with a window size equals 5. Then \( x, y \) coordinates are resampled into 30 points with bilinear interpolation. The resampling rate could be controlled by the Nyquist Sampling rate to achieve improved Signal Noise Ratio (SNR). Scales of the characters are normalized during resampling. Turning function
is defined as direction angles alongside boundaries. Four-quadrant arctangent is used to compute $\Theta(s)$. And signs of $\delta x$ and $\delta y$ are used to determine the quadrant. The value range of $\Theta(s)$ is $[-\pi, \pi]$ while the value range of regular arctangent is $[-\pi/2, \pi/2]$.

$$\Theta(s) = \arctan\left(\frac{\delta y(s)}{\delta x(s)}\right) \quad (5.1)$$

As shown above, tangent angle values are between $-\pi$ to $\pi$. Angle jump happens when reaching the limits. For example, if previous point is $0.9\pi$ and current point is $1.1\pi$, the increment should be $0.2\pi$. But because of the value limits, current values is changed to $-0.9\pi$, and increment becomes $-1.8\pi$. To deal with this problem, angle unwrapping is introduced. The increment is computed for each pair of points in turning functions, if it is greater than $\pi$, the current value is added or subtracted with times $2\pi$, until smallest increment is found. Turning function values could be arbitrary large or small after angle unwrapping.

### 5.2 Classification

#### 5.2.1 Alignment

Shifting start point $O$ along the perimeter of the symbol by an amount $t$ causes a horizontal shift $\Theta_A(s + t)$ (as shown in Figure 5.2a). Rotating of a symbol image of $A$ counterclockwise by $\theta$ causes turning function shifting vertically to $\Theta_A(s) + \theta$ (as shown in Figure 5.2b).

In an image with noise, turning functions are affected by horizontal and vertical shift simultaneously. To compare two turning functions, a method to search for their smallest distance by searching horizontally and vertically simultaneously is proposed. This alignment method enables us to compare two turning functions meaningfully, as we know turning functions with different starting point are circular shifts of the others.

First, two copies of one turning function (turning function of training symbol in our system) are concatenated to form a new function with double length. Then unwrap is applied towards both the duplicated turning
function and the other turning function (turning function of testing sample). Next, we search the duplicated turning function in a sliding window and each window is of size 30 points. The window truncated part of the double turning function and the testing turning function is compared, in terms of Pearson’s correlation as shown in Equation 5.2.
Figure 5.2: Horizontal and Vertical shift of turning function due to starting point movement and rotation. (a) Horizontal shift of turning function by shifting starting point; (b) Vertical shift of turning function by rotation.

\[ \rho(x, y) = \frac{cov(X, Y)}{\sigma_x \sigma_y} = \frac{E[(X - \mu(x))(Y - \mu(y))]}{\sigma_x \sigma_y} \]  

(5.2)

where \( cov(X, Y) \) is the covariance of \( X \) and \( Y \), \( \sigma \) is the standard deviation, and \( \mu \) is the mean value.

Therefore, the vertical shift is compensated, because correlation computation requires each function deducted by their average. The correlation will
be measured for each sliding window, and the maximum correlation value is used as a similarity measurement of two turning functions. An example of matching result is shown in Figure 5.3. The time complexity for each pair of features is $O(n^2)$, where $n$ is the number of points in a turning function.

Finally, we build a cost function, which measures the difference between two symbols, while compensating for scale, rotation, translation and other non-rigid transformation, using turning functions. The maximum correlation will be computed for each pair of training sample and testing sample. And the label of the training instance with largest correlation will be used as the prediction label for the testing sample.
5.2.2 Curvature Histogram

If the robustness of the classifier is to be increased, a larger training dataset is necessary. However, it will add to the computational cost, as the nearest neighbor classifier computation time is linear to the number of the samples in training set. To remedy the problem, the histogram of the curvature function is used instead of the turning functions. This way, alignment procedure can be avoid and other advanced classification methods can be utilized, e.g. Support Vector Machine (SVM).

The turning function is doubled and concatenated to form a duplicated turning function as we described earlier. This way, break points of the turning functions can be compensated for. Next, the same angle unwrapping method is utilized to remove angle jumps at $-\pi$ or $\pi$. Because the angle values can increase or decrease without limit after angle unwrapping, the absolute value of the turning function becomes unsuitable for direct comparison. The curvature histogram can also circumvent this issue.

The first derivative of the turning function is computed using Equation 5.3. The derivative step removes the effect of the arbitrary angle value increase or decrease, as one do not care about the absolute value of the angles, but the shape of the angles. By computing the first derivative, the local curvature at each point of the contours instead of the azimuth angle of the contour is actually used.

\[
d\Theta(s) = \Theta(s) - \Theta(s - 1)
\]  

(5.3)

However, as the starting point of the contour extraction is not perfect, the curvature function can’t be used directly for comparison. Since the histogram of the function is used, no matter what the starting point is, one can always get robust representation of the shape of the symbols, because histograms are not position dependent.

The histogram will then be calculated for the duplicated curvature functions for each symbol. The value range of the histogram will be fixed to $[-\pi, \pi]$. Notice that there should be no value exceeds that range, as the curvature function is representation of the boundary angle variations, and therefore the variation at any point cannot exceed 180 degrees.
The histogram of curvature function is computed using 30 bins, and the final feature for each symbol is a 30 point vector. Since alignment step is not required, a SVM classifier can easily be used. The classifier is a ‘RBF’ (Gaussian) Kernel SVM. The soft margin parameter $C$ is set to 1.0. It is trained on $INFTY$ training dataset. The classifier is trained on 10,000 samples from the training set, then used to recognize the characters extracted from license plate images.
Chapter 6

Experiment Result and Analysis

6.1 Dataset

In experiment, MediaLab LPR database is used as testing images, and character recognition training dataset is acquired from INFTY dataset.

6.1.1 MediaLab Dataset

The LPR database is provided by MediaLab of National Technical University of Athens (NTUA) [2]. The database contains images in BMP and JPEG format with license plate in natural scenes mounting on the front or back mobile vehicles. The license plates are European, which contain black character on white background. The characters’ fonts are sans-serif. The characters are usually evenly distributed from left to right on the license plate with one single text line. However, in some special customized license plate, the characters are not evenly distributed with random spaces between characters.

The license plates contains permutation of English and numerical symbols. Usually, there will be two or three English characters at the left hand side of the license plate, and there will be four numeric characters on the right hand side of the license plate. The English and numeric characters are separated with figure dash. The dataset contains three categories, still images, extremely difficult cases and videos. Still images set are used in our experiment.

The still image category contains 7 different cases, which are:

- Case 1: Daytime color images, which are captured in daylight with only one license plate in each image. The license plates are usually in the center of the images.
• Case 2: Daytime grey scale images. Other properties are similar to case 1.
• Case 3: Images with blurring, usually motion blurs caused by unsteady camera.
• Case 4: Images with shadows. Therefore, license plates are not evenly illuminated.
• Case 5: Close view, which contains larger license plate. This usually makes the license plate easier to be detected and recognized.
• Case 6: Daytime color images (large), where the content of the image is similar to case 1, only the image size is larger;
• Case 7: Night capture, which are captured at night. Because of the use of flashlight, license plates are all well illuminated. Usually, they are easier than daytime images.

The number of image in each case is listed in Table 6.1. The experiment contains two major components, detection and recognition. For detection, the images sizes are normalized in order to accelerate the procedure. Since larger images are 1792 by 1312 pixels and normal images sizes are 800 by 600 pixels, all are resized to 800 pixels in width, while maintaining the aspect ratio.

### 6.1.2 INFTY Dataset

**INFTY** dataset was established in **INFTY** project, containing English characters, numeric characters, Greek letters and mathematical symbols [34]. This dataset is used for training our system, because it contains a sufficient quantity of individual characters with labels.

**INFTY** dataset contains three categories of images, including **INFTYCDB1**, **INFTYCDB2** and **INFTYCDB3**. As **INFTYCDB1** and **INFTYCDB2** contain mathematical document images, they are not useful for our experiment. The **INFTYCDB3** contains individual symbols including both English and numeric characters. So this dataset is used for our training. The **INFTYCDB3** dataset contains two groups, **INFTYCDB3A** and **INFTYCDB3B**, for training and testing purposes respectively. Training set contains 188, 752
different images containing individual symbols, while testing set contains 70,637 symbols.

To reduce the training set, we select only 10,000 training images from INFTYCD3B3A. The training set contains only alphanumeric symbols. By removing symbols that does not contained by typical license plate, such as mathematical operators, brackets, etc. We improved the recognition accuracy while reducing the computational time.

### 6.2 Ground Truth and Evaluation Metric Design

The license plates region bounding box ground truth is labeled manually, using rubber band rectangle GUI tools developed by us. The bounding box of license plates is marked, and coordinates of bounding box corners are recorded for comparison with detection results. The area of the ground truth, detection region and the overlapped region are used to compute the Precision, Recall and F-metric of the system. The Precision, Recall and F-metric are defined as in Equation 6.1, 6.2 and 6.3.

The precision is defined as the percentage that retrieved instances that are relevant. The recall is defined as the percentage that relevant instances that are retrieved. And F-measure is the harmonica mean of the precision and recall.

\[
\text{precision} = \frac{\text{ground truth area} \cap \text{prediction area}}{\text{prediction area}} \quad (6.1)
\]

\[
\text{recall} = \frac{\text{ground truth area} \cap \text{prediction area}}{\text{ground truth area}} \quad (6.2)
\]

\[
\text{F-measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (6.3)
\]

The recognition labels are compared with the ground truth labels. The recognition system produces Unicode of characters. The recognition performance is evaluated using accuracy. The accuracy is defined as the percentage of correctly classified samples in all recognition testing samples.
6.3 Detection Performance Evaluation

The images are tested on all images from the MediaLab dataset as shown in Table 6.1. The Precision of our system is 87.68%, the Recall is 91.30%, and F-measure is 89.45%. This metric considers the pixel mismatching between the ground truth and the predictions. Because for some license plates, the detected region is larger or smaller than the labeled ground truth, it is not 100% correct. The precision and recall value will be based on the pixels overlapping between the two regions, even the detection is considered correct by human evaluator.

The errors of our system are caused by many sorts of reasons. The texts in the background would be recognized as the license plate if they have similar patterns and colors. For example, a shop sign or a traffic sign would look similar as a license plate, and they generate false positive samples. The other reason is that the detected region of interest doesn’t match exactly to the true license plate region. It might be larger or smaller than the license plate, because of noise, occlusion and viewing angles. Some are solvable problems for our system. For example, the recognition will generate a very low confidence for additional objects, and one could eliminate those regions by checking the recognition confidence. Some characters are missed, where the license plate was not fully detected. Some examples are shown in Figure 6.2.

From all testing results, only one image totally missed the license plate. The image is severely blurred, and the license plate is extremely difficult to
Figure 6.2: Example output of license plate detection system. Some error detection examples are shown, where license plate is missed or detection region is larger than true region of license plate.

be detected. The precision, recall values are very high, despite all the difficult cases and strictness of our evaluation metric. The large images have better performance than the small images, mainly thanks to better resolutions. The blurred images get worse performance, due to the difficulties of extracting correct CCs. The shadow images sometimes have part of the license plate detected while another half is missed. In those cases, although adaptive binarization suppressed errors caused by uneven illumination, problem remains when the shadow is too dark, or contrast is too large. The night images, on the other hand, are easily detected, because they are always well illuminated by flashlight of the camera. The performance of our detector for different categories of the images are listed in Table 6.2.

Table 6.2: Precision, Recall and F-metric of the license plate detection system.

<table>
<thead>
<tr>
<th>Category</th>
<th>Color</th>
<th>Grayscale</th>
<th>Large</th>
<th>Blurred</th>
<th>Shadow</th>
<th>Close</th>
<th>Night</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>82.54</td>
<td>93.26</td>
<td>90.11</td>
<td>65.30</td>
<td>85.41</td>
<td>76.77</td>
<td>90.67</td>
<td>87.68</td>
</tr>
<tr>
<td>Recall</td>
<td>92.50</td>
<td>91.09</td>
<td>94.01</td>
<td>85.40</td>
<td>92.31</td>
<td>89.12</td>
<td>91.01</td>
<td>91.30</td>
</tr>
<tr>
<td>F-metric</td>
<td>87.24</td>
<td>92.16</td>
<td>92.02</td>
<td>74.00</td>
<td>88.73</td>
<td>82.48</td>
<td>90.84</td>
<td>89.45</td>
</tr>
</tbody>
</table>
6.4 Character Segmentation and Recognition Evaluation

The detected license plates are then segmented and recognized. The segmentation is performed using FPV technique, while recognition is performed using turning function based techniques. Some examples of recognized license plates are shown in Figure 6.3. In case that the characters are touched or broken, correct recognition labels could still be get. The touched characters are separated by FPV method using projection profile, as characters ‘4’ and ‘6’ in Figure 6.3a. Turning function based recognition method has good robustness against missing parts of characters, as character ‘M’ in Figure 6.3c.

However, due to the strong noises, variances and deformations in testing datasets, there are some wrong segmentations and miss-recognitions. Some examples are shown in Figure 6.4. As in Figure 6.4(a), the license plate image is so heavily blurred, even human eyes can not recognize characters. Our system failed in segmentation or recognition, but successfully detect the correct location. In Figure 6.4(b), some characters are miss-recognized due to strong blurring, but segmentation is correct. In Figure 6.4(c), the image is heavily skewed. We successfully rotate the image back to the right position and generate the correct segmentation result. The last character is miss-recognized due to noise and touching with the boundary. In Figure 6.4(d), the license plate is wrongly segmented, because the license plate is too small and characters have about only 1 pixel stroke width. The wrong segmentation also affects the final recognition result, where 2 out of 7 characters are recognized wrongly. In Figure 6.4(e), one characters is miss-recognized due to touching. In Figure 6.4(f) and (g), characters are miss-recognized due to heavy skewing and rotation. In case of strong luminance inhomogeneity, we manage to get very robust performance, as shown in Figure 6.4(h). Even the license plate is covered by strong shadow edges, we could still extract correct binary map of the image. This is because of the locally adaptive threshold we used.

The recognition accuracy is listed in Table 6.3. The color images, grayscale images and large images get very similar recognition accuracy, while blurred
Figure 6.3: Example output of license plate recognition system. The detected region (as shown in second row) is firstly binarized using adaptive thresholding, and projection profile and FPV is used to segment binary image (third row) into individual characters. The output labels are shown in captions below image pairs.

Images and shadow images get low accuracy. The blurred images are difficult to process, because boundaries are difficult to be extracted. The shadow images produce characters that are partially cropped. The shadow lines sometimes cut the characters into two segments, which cannot be recognized correctly by our system. The night images, again, are very easy to be recognized, as they are well illuminated by the flashlight.

To evaluate segmentation performance, license plates are cropped out using ground truth bounding boxes, segmentation algorithm with FPV are performed and then compare with ground truth. If two characters are segmented into one (under-segmentation), one character is segmented into two (over-segmentation), or cutting point is miss-aligned with inter-character white space (wrong segmentation), the correct segmentation count will be subtracted by 1. Then total correct segmentation is divided by total number of segments to compute segmentation accuracy, as shown in Table 6.3. The total accuracy for segmentation is 97.54%.

To evaluate recognition accuracy, each individual character is cropped out and recognized using turning function based algorithm. The predict labels is then compared with the ground truth labels, then recognition accuracy is computed, as seen in Table 6.3. The total accuracy for individual
character recognition is 94.20%.

For overall performance, the license plate character labels for all testing images are used as ground truth. Characters that missed by the localization could not be recovered by the recognition component. On the other hand, some non-text objects detected in the localization step become false alarms. Similarly, segmentation and recognition rates will also affect the final overall performance. Errors in each step of the pipeline will accumulate and add
to the final error rate. Therefore, the accuracy showing in Table 6.3 overall is lower than an isolated recognition component performance.

Table 6.3: Accuracy for segmentation, recognition and overall system for different categories of images. Segmentation accuracy is computed when assuming all license plates are correctly detected as ground truth bounding boxes; recognition accuracy is computed when assuming all characters are correctly segmented. The overall accuracy is computed when combining proposed localization, segmentation and recognition systems. So the errors will accumulate through each stage of the system.

<table>
<thead>
<tr>
<th>Accuracy(%)</th>
<th>Color</th>
<th>Grayscale</th>
<th>Large</th>
<th>Blurred</th>
<th>Shadow</th>
<th>Close</th>
<th>Night</th>
<th>Total(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segmentation</td>
<td>99.50</td>
<td>99.45</td>
<td>98.60</td>
<td>89.70</td>
<td>86.80</td>
<td>98.85</td>
<td>100</td>
<td>97.54</td>
</tr>
<tr>
<td>Recognition</td>
<td>94.32</td>
<td>95.67</td>
<td>98.90</td>
<td>50.22</td>
<td>85.77</td>
<td>96.67</td>
<td>95.00</td>
<td>94.20</td>
</tr>
<tr>
<td>Overall</td>
<td>92.33</td>
<td>91.11</td>
<td>90.18</td>
<td>33.19</td>
<td>69.74</td>
<td>89.53</td>
<td>95.77</td>
<td>87.33</td>
</tr>
</tbody>
</table>

The turning function and curvature histogram based methods are compared. The method of utilizing turning function and curvature histogram is introduced in Section 5. The turning function method achieved better performance, while the curvature histogram gets significant advantages on speed. The SVM classifier for curvature histogram is slow to train, but very fast in execution. On the other hand, turning function requires alignment procedure, and similarities are computed between each pair of training and testing samples, therefore the speed is much slower. The final system could choose to use turning function or curvature histogram method based on whether accuracy or speed is a priority.

Combining the turning function and the curvature histogram in the future could also be possible. The SVM classifier could provide a multi-class prediction, where the firstly several class with largest confidences are recorded. Then one could use turning function method and search only within candidate classes, which are suggested by SVM classifiers. Therefore, one could manage to design a system with faster speed than turning function, while getting higher accuracy than curvature histogram.

The performance comparison between turning function and curvature histogram based method is shown in Table 6.4, as tested on different categories of the NTUA Media Lab dataset. The turning function has higher accuracy than curvature histogram, as it preserves more information of the shape of the character. However, curvature histogram is significantly faster.
than turning function as it doesn’t need alignment.

Table 6.4: Accuracy for different categories of images using turning function (TF) and curvature histogram (CH) based SVM.

<table>
<thead>
<tr>
<th></th>
<th>Color</th>
<th>Grayscale</th>
<th>Large</th>
<th>Blurred</th>
<th>Shadow</th>
<th>Close</th>
<th>Night</th>
<th>Total(%)</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF</td>
<td>92.33</td>
<td>91.11</td>
<td>90.18</td>
<td>33.19</td>
<td>69.74</td>
<td>89.53</td>
<td>95.77</td>
<td>87.33</td>
<td>25.79</td>
</tr>
<tr>
<td>CH</td>
<td>91.15</td>
<td>90.56</td>
<td>88.98</td>
<td>46.96</td>
<td>61.67</td>
<td>87.44</td>
<td>91.28</td>
<td>85.50</td>
<td>6.74</td>
</tr>
</tbody>
</table>

### 6.5 Performance Comparison

To evaluate the proposed system, we compare its overall performance with other existing LPR systems.

Systems are proposed in recent years to recognize Chinese [40, 23, 37, 25, 13, 36, 45, 24], Arabic [26, 6] and other types of license plate, and achieved very appealing accuracies. Some existing systems’ performance are listed in Table 6.5. However, because LPR systems are developed to process different kinds of license plates in different countries under different conditions and for different purposes, most of the previous systems used their own datasets for evaluation. The reason is mainly due to that LPR systems are very practical, each system tends to use images captured under its specific environment to maximize the performance for particular purpose. This makes performance comparison difficult.

To get a fair comparison, we used a publicly available dataset from MediaLab [2]. So we could compare our performance with Anagnostopoulos’ system [3] in terms of localization and recognition. As in Table 6.6, the proposed system achieved better accuracy in terms of localization and much improved accuracy in overall system recognition rate.

The proposed system also gets fast execution. The computation time comparison is listed in Table 6.7. Notice that total elapse time is not the summation of localization and recognition, because preprocessing and segmentation time are not included in the table. Our system is tested using Matlab 2012a on MacBook pro with Intel Core 2 Duo at 2.4 GHz, with 512 MB RAM. As Matlab is not known for its speed, the proposed system could be further accelerated using compiled languages like C++. 
Table 6.5: Existing license plate recognition systems, which used different testing datasets and could not be compared directly with proposed system.

<table>
<thead>
<tr>
<th>System</th>
<th>Dataset</th>
<th>Localization (%)</th>
<th>Recognition (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wu [40]</td>
<td>Huyonghang highway charging station</td>
<td>100</td>
<td>94.8</td>
</tr>
<tr>
<td>Mohammad [26]</td>
<td>Miser University Arabic License plate</td>
<td>92</td>
<td>93</td>
</tr>
<tr>
<td>Lee [23]</td>
<td>Self-collect dataset</td>
<td>-</td>
<td>97</td>
</tr>
<tr>
<td>Wan [37]</td>
<td>License plate characters</td>
<td>-</td>
<td>94</td>
</tr>
<tr>
<td>Ling [25]</td>
<td>Self-collect dataset</td>
<td>-</td>
<td>96.7</td>
</tr>
<tr>
<td>Gao [13]</td>
<td>Self-collect traffic surveillance image</td>
<td>96.5</td>
<td>89.9</td>
</tr>
<tr>
<td>Tian [36]</td>
<td>LP dataset [46]</td>
<td>92.4</td>
<td>-</td>
</tr>
<tr>
<td>Zhao [45]</td>
<td>Self-collect all-weather images</td>
<td>97.7</td>
<td>95.6</td>
</tr>
<tr>
<td>Li [24]</td>
<td>Self-collect surveillance image</td>
<td>96.0</td>
<td>91.9</td>
</tr>
</tbody>
</table>

Table 6.6: Proposed system performance comparing to Anagnostopoulos’ system using NTUA MediaLab dataset [2].

<table>
<thead>
<tr>
<th>System</th>
<th>Dataset</th>
<th>Localization (%)</th>
<th>Recognition (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anagnostopoulos [3]</td>
<td>MediaLab Dataset</td>
<td>89.1</td>
<td>86.0</td>
</tr>
<tr>
<td>Proposed</td>
<td>MediaLab Dataset</td>
<td>89.45</td>
<td>87.33</td>
</tr>
</tbody>
</table>

Table 6.7: Proposed system computation time comparing with Anagnostopoulos’ system.

<table>
<thead>
<tr>
<th>System</th>
<th>Localization (ms)</th>
<th>Recognition (ms)</th>
<th>Total (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anagnostopoulos [3]</td>
<td>117</td>
<td>128</td>
<td>276</td>
</tr>
<tr>
<td>Proposed System</td>
<td>35.01</td>
<td>25.79</td>
<td>102.33</td>
</tr>
</tbody>
</table>
Chapter 7

Conclusion

We proposed an end-to-end system for license plate localization and recognition. Image Noises are reduced by bilateral filters. The system utilizes adaptive thresholding techniques to reduce the impact of uneven luminance to find the location of the license plate. Candidate objects are filtered with a two-step condition checking method, where character shape information and their spatial relationship information is used to extract correct objects as candidate license plate regions. Turning function is used to convert a 2-D image matrix into a 1-D feature vector, while preserving the shape information. The turning function method is robust against rotation, scaling and translation. The curvature histogram is used along with turning functions to classify the individual symbols. The performance using these two different kinds of features are compared. Each of them has their own advantages. The turning function method provides higher accuracy, while the curvature histogram has much better speed in execution. We test our system using INFTY symbol image dataset for training and MediaLab license plate image database for testing. A Graphical User Interface (GUI) is created for both ground truth labeling and license plate detection, allowing user to utilize our system practically. The proposed system achieved 89.45% for localization and 87.33% for overall recognition rate. This performance is compatible with state-of-art systems using same testing set.

In the future, a method for license plate detection using machine learning algorithm will be developed. Therefore, a recognition system can be automatically trained, once sufficient labeled datasets are provided. It is more robust against variation of character sizes and degradations.

Turning function and curvature histogram could be combined to increase
accuracy while decreasing computational time. A possible model utilizes curvature histogram to get an initial estimate and uses a following turning function method for accurate classification.

Images in extremely difficult cases will be considered, where severe blurring (either motion blurring or Gaussian blurring), low signal noise ratio and highly uneven illumination will be addressed. And dedicated license plate font database could be more convenient for training a recognition system.
Bibliography


[6] Raja Bala, Yonghui Zhao, Aaron Burry, Vladimir Kozitsky, Claude Fil-
lion, Craig Saunders, and José Rodríguez-Serrano. Image simulation
for automatic license plate recognition. volume 8305, pages 83050Z–
83050Z–9, 2012.

images using radon transform based projection profile. In Soft Comput-
ing and Pattern Recognition (SoCPaR), 2011 International Conference

network-based license plate recognition system. Vehicular Technology,

[9] Shyang-Lih Chang, Li-Shien Chen, Yun-Chung Chung, and Sei-Wan
Chen. Automatic license plate recognition. Intelligent Transportation

[10] P. Comelli, P. Ferragina, M.N. Granieri, and F. Stabile. Optical recog-
nition of motor vehicle license plates. Vehicular Technology, IEEE

nology for toll violation enforcement. In Image Analysis for Transport
Applications, IEE Colloquium on, pages 7/1–7/5, Feb 1990.


[13] Jianjun Gao, Erik Blasch, Khanh Pham, Genshe Chen, Dan Shen, and
Zhonghai Wang. Automatic vehicle license plate recognition with


[26] Khader Mohammad, Sos Agaian, and Hani Saleh. Practical automatic


