Scientific Bulletin of the „Petru Maior" University of Tîrgu Mureş
Vol. 10 (XXVII) no. 1, 2013
ISSN-L 1841-9267 (Print), ISSN 2285-438X (Online), ISSN 2286-3184 (CD-

AUTOMATIC NUMBER PLATE RECOGNITION SYSTEM FOR IPHONE DEVICES

Attila ASZALOS[1], Călin ENĂCHESCU[2]
"Petru Maior" University of Tîrgu Mureş
Nicolae Iorga street, no.1, Tîrgu-Mureş, Romania
[1]ecalin@upm.ro
[2]aszi.atti@hotmail.com

Abstract

This paper presents a system for automatic number plate recognition, implemented for devices running the iOS operating system. The methods used for number plate recognition are based on existing methods, but optimized for devices with low hardware resources. To solve the task of automatic number plate recognition we have divided it into the following subtasks: image acquisition, localization of the number plate position on the image and character detection. The first subtask is performed by the camera of an iPhone, the second one is done using image pre-processing methods and template matching. For the character recognition we are using a feed-forward artificial neural network. Each of these methods is presented along with its results.

Keywords: image processing, number plate recognition, edge detection, optical character recognition, mobile devices, iOS

1. Introduction

Automatic number plate recognition (ANPR) systems are used to detect and recognize number plates on images obtained by cameras. This task which is trivial for a human is a much more complex task for a computational system. These systems are composed of two main parts: hardware and software part [2]. The hardware part consists of image acquisition equipment, which is often optimized to capture clear images even in low light and bad weather conditions. The captured images are then processed by the software part of the ANPR system, which provides the recognized characters on the number plate.

There are two types of ANPR systems: static and dynamic. Static ANPR systems recognize the number plate of still vehicles, while dynamic systems recognize the number plate for moving vehicles. Some systems are capable for the recognition of multiple license plates on a single image.

Our ANPR system uses as hardware the camera of an iPhone 4 device, running the iOS 5 operating system. The software part is implemented for the iOS platform using the C and Objective-C languages. It is a static ANPR system, detecting the number plate for still vehicles and can detect only a single license plate on an image. This system was implemented as a static library, because this way it can be easily integrated into any iPhone application. Our main goal was to build an ANPR system which can detect number plates in real-time on an iPhone device, which has low hardware resources compared to existing ANPR systems.

The structure of paper is the following: the next section presents similar systems, compares them with our system and discusses the similarities and differences. The third section describes the architecture of our system and the role of each component. The fourth section presents the used image processing and OCR methods, and how these are combined to solve the problem of automatic number plate recognition. The fifth section will present the recognition results, for each component in part and execution time measurements. The last two sections contain the conclusions and references.

2. Feature comparison with a similar system

Before we started to build our system we have conducted a research for similar existing systems running on iPhone devices. We have found such a system built by Roman Sládeček for the bachelor's thesis entitled "iPhone Application for Number Plate Recognition", supervised by eng. Boris Procházka at Brno University of Technology, Czech Republic in 2011 [14]. In the following we will compare the

features of this system with the features of our system.
The first difference we have found is in the fact that our system is built as a static library, while Sládeček's system is built as an iPhone application. A static library can be integrated more easily into a new application than the extracted source code from an application.
A second feature we would like to compare is the source type for the ANPR system. Sládeček's system requires as input an image file, either from the phone's storage or from an image captured with the camera. Our system can recognize a number plate on image files, and even more than that it can work on real-time images obtained from the video camera, and detect the location of the number plate in real-time.
A third difference is regarding the location of the number plate. Sládeček's system requires that the number plate is located in the center of the image. Even a rectangle is drawn on the screen, where the number plate should be positioned, when the user is taking a photo with the camera. This is a disadvantage because the number plate can't be always positioned in this predefined area. Our system doesn't require a specific location on the image for the number plate, since it can locate it automatically.
In case of Sládeček's system when the plate image is detected it has to be adjusted by the user so the characters are positioned at predefined positions. Our system detects the location of the characters automatically.
In the following sections we will refer back to Sládeček's system, regarding the algorithms and image processing methods used.

3. System architecture

The system we have built is an iPhone application, which can be used for automatic number plate recognition. The architecture of the system can be seen on Figure 1.

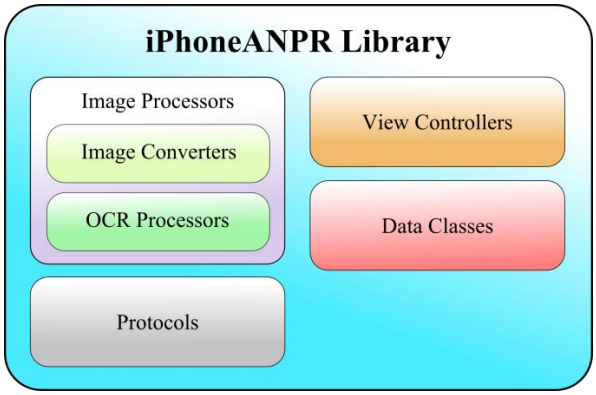

Figure 1 – System architecture

The system is split into 4 main parts:

1. Image processors
2. Protocols
3. Data classes
4. View Controllers

3.1. Image processors

The image processing classes use the open source OpenCV library to perform various image processing operations [8]. These classes work with the images captured by the camera of the phone. The image processing methods and the sequence of these will be described in the following section.

This part also contains image converter classes, which perform type conversions between the image objects from the iOS Foundation framework and the OpenCV IplImage object types.

The OCR processors are used to recognize the characters on the number plate image. Our system has 2 character recognition engines. The first one is an engine based on an artificial neural network, created and trained by us. The second one is the open source Tesseract OCR engine [15].

3.2. Protocols

The defined protocols define the type of messages used by the image processors to notify the interested objects about the results of the processing.

3.3. Data classes

The data class type objects encapsulate the information about a number plate. This information mainly consists of the number plate image and the recognized string on the number plate.

3.4. View controllers

The role of the view controllers is to interact with the user, present the user interface elements and recognize the number plate using the previously presented parts of the system. The user interface contains a part where the user can see the input from the phone's camera. If on an input frame the number plate is found a rectangle is drawn around it, making the feeling of an augmented reality application.
The following section describes the image processing methods used by our system. It also describes our OCR engine, the structure of it, how it was trained and how does it work.

4. Description of the methods

The problem of automatic number plate recognition can be split into the following steps: image acquisition, determination of the number plate location and character recognition.

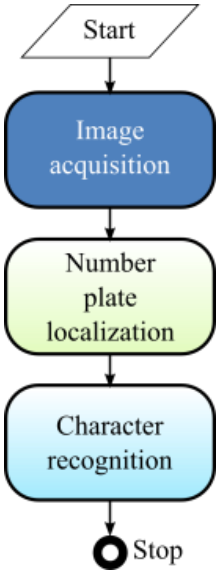

Figure 2 – ANPR schema

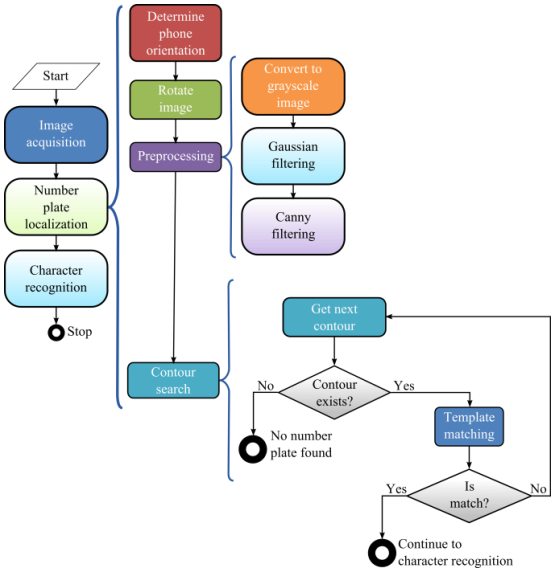

Figure 3 – Number plate localization

4.1. Image acquisition

Our system uses the camera of an iPhone and samples the video output of the camera. The captured frames have a medium size, which according to the documentation from Apple is suitable for sharing over WiFi. The exact size for a frame is 360 x 480 pixels [3]. The captured images are RGB color images with 3 channels.

4.2. Determination of the number plate location

The first step in the localization of the number plate is to determine the orientation of the device. An assumption is made about the car of which number plate is about to be recognized. The assumption is that the car is on its wheels and not upside down. This way we can rotate the image to the "correct" orientation where the number plate is positioned horizontally on the image. This way the following steps of the algorithm don't have to deal with the case in which the number plate is positioned vertically. This rotation is required because if we hold the device in landscape mode the captured image will also be in landscape mode [4], with the number plate oriented vertically.

After rotating the image to the "correct" orientation starts the preprocessing of the captured image (frame). First the color image is converted to a grayscale image. This is followed by a Gaussian filtering operation of the grayscale image. This filtering improves the result of the next operation, the Canny filtering [12]. The result of the Canny filtering is a binary image containing the edges [10]. On this edge image a contour search is started. For every found contour there exists a bounding rectangle [11]. Using the coordinates and size of this bounding rectangle the algorithm makes a template matching. The used template is an edge image of a number plate. The template is resized to the size of the contour's bounding rectangle and is positioned exactly over it. For the template matching the following formula [7] is used, with the value in $R(0,0)$:

$$R(x,y) = \frac{\sum_{x',y'}\left(T(x',y') \cdot I(x+x',x+y')\right)}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

The result of the template matching is a number which reflects the similarity between the template and the content in the image surrounded by the bounding rectangle. This resulting value is compared to a predetermined threshold value. This threshold value was determined after conducting a series of experiments. If the result of the template matching is greater than the threshold value we have found a match. If there are multiple matches the one with the highest match value is taken.

At the end of this number plate localization algorithm the location and size of the number plate is determined.

4.3. Character recognition

Our character recognition algorithm takes as input the captured color image and the location and size of

the number plate. In the first step the color image is preprocessed again, in a different way from the preprocessing from the number plate localization algorithm. The first step of this preprocessing is an adaptive filtering, which eliminates the illumination variations on the image [6].

After the preprocessing the number plate portion of the image is extracted. This algorithm recognizes the characters individually. This means they have to be separated. In order to separate the characters of the number plate image we have used the technique of image projections. This way the horizontal and vertical boundaries of the characters can be found on the number plate image [1]. The result of the vertical projection is a discrete function of which there can be determined the maximum points. The coordinates of the maximum points give the position of the boundaries between the characters. The following pseudo-code algorithm describes this maximum search [13]:

1. Find the x coordinate for the maximum value in the projection $x_{max} = \text{pos}\{\max_{0 \le x \le w}\{P(x$
2. Determine the width of the peak
   a. $x_{left} = \text{pos}\{\max_{x \le x \le x_{max}}\{x | P(x) \le c_x \cdot P(x_{max})\}\}$
   b. $x_{right} = \text{pos}\{\min_{x_{max} \le x \le w}\{x | P(x) \le c_x \cdot P(x_{max})\}\}$
3. On the $[x_{left}, x_{rig}$ interval we set all values to
4. If $P(x_{max}) < c_w \cdot v_n$ the algorithm stops
5. At x_r coordinate we have a boundary between two characters
6. The algorithm continues from step 1.

The next step in the algorithm is to determine the vertical boundaries of the characters. For this the technique of the horizontal projections can be used. The following algorithm is used to determine the vertical boundaries of the individual characters:

1. Detect the vertical centre of the character image: y_m :
2. Determine the top margin of the character: $y_{top} = \text{pos}\{\max_{0 \le y \le y_m}\{P(y$
3. Determine the bottom margin of the character: $y_{bottom} = \text{pos}\{\min_{y_m \le y \le h}\{P(y$

At this point the character images from the number plate are extracted. To recognize the individual characters we have used an artificial neural network. The structure of the network can be seen on Figure 4 [5].

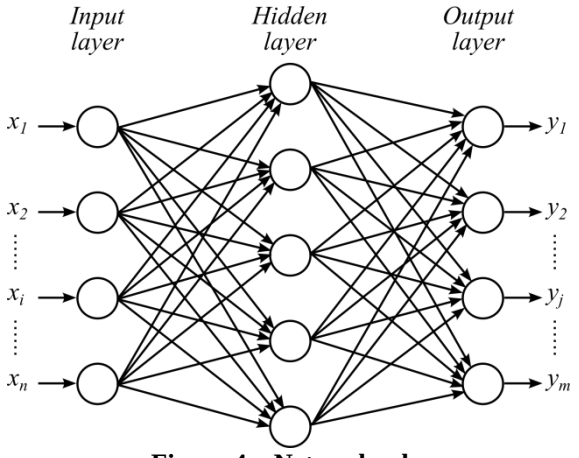

Figure 4 – Network schema

The input layer has 8 inputs, which correspond to eight statistical indicators calculated from each individual character image. The hidden layer has 15 neurons and the output layer has 36 neurons. The outputs take values from -1 to +1, indicating if the character isn't or is recognized.

A	B	…	Z	0	…	9
0	1	…	25	26	…	35

Table 1 – Output coding of characters

In the following table we summarize the statistical indicators used as input values for the neural network [9].

Name	Formula
Center of gravity	$CG = \frac{\sum_{i=0}^{n} i \cdot P(i)}{\sum_{i=0}^{n} P(i)}$
Skewness	$SK = \frac{\mu_3}{\sigma^3} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{\left(\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{N}\right)^{3/2}}$
Kurtosis	$KU = \frac{\mu_4}{\sigma^4} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4}{\left(\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{N}\right)^2}$
Orientation	$\phi = \frac{1}{2}\arctan\frac{2 \cdot \mu_{1,1}}{\mu_{2,0} - \mu_{0,2}}$
Excentricity	$\varepsilon = \frac{\sqrt{a^2 - b^2}}{a} = \frac{\left(\mu_{2,0} - \mu_{0,2}\right)^2 - 4 \cdot \mu_{1,1}^2}{\left(\mu_{2,0} - \mu_{0,2}\right)^2}$

Table 2 – Statistical indicators

The network has been trained with 490 samples, collected with an iPhone running the plate localization algorithm and separating the character images. This way the network was trained with images from the real world. To recognize a character from the number plate we calculate the statistical indicators and feed them to the input of the neural network. From the output we take the index of the closes value to +1.

5. Results

To measure the effectiveness of our system we have conducted 3 types of tests: efficiency of number plate localization, efficiency of character separation and efficiency with character recognition. The results of these tests are shown on the following charts.

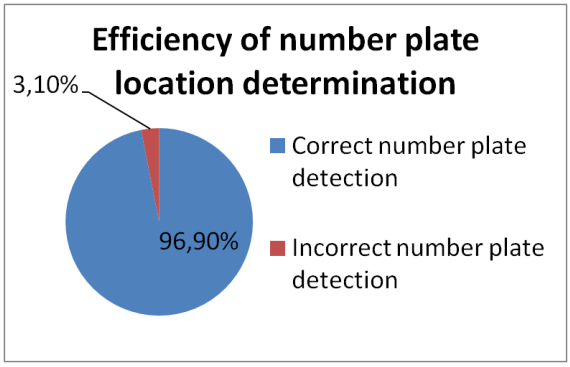

Figure 5 – Number plate detection results

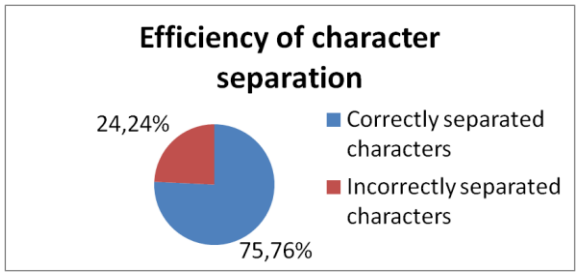

Figure 6 – Character separation results

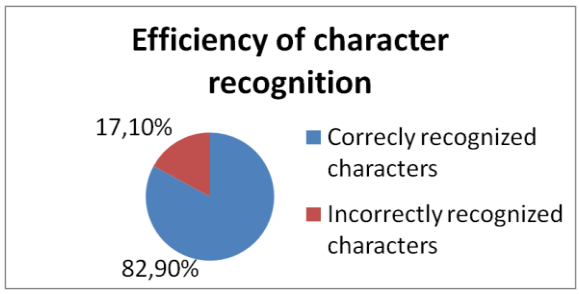

Figure 7 – Results of character recognition

Figure 8 shows the character recognition success for the individual characters.

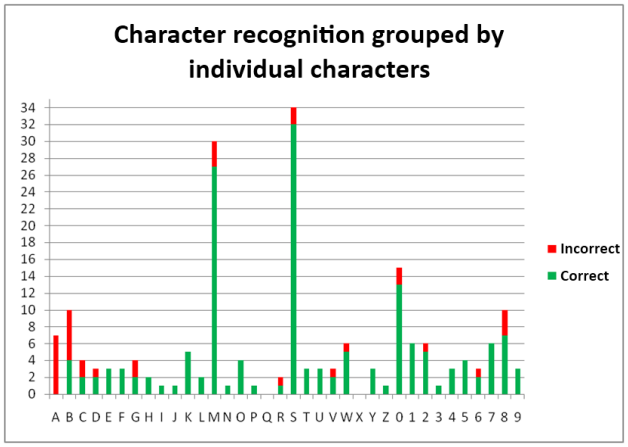

Figure 8 – Individual character recognition results

We have to state that these are the results for a relatively small training set and they should increase along with the size of the training set.

6. Conclusions and future work

As conclusions we can state that our system has the following advantages over the system built by Roman Sládeček:

- For the number plate recognition no user action is required, our system is fully automatic
- The location of the number plate is detected and our system doesn't require to be in a predefined location
- The character separation is done automatically

In the future we would like to improve the accuracy of the character separation and the character recognition. We are planning to gather more data so we can increase the size of the training and testing sets used by the neural network.

7. References

[1] Anil K. Jain, Fundamentals of Digital Image Processing, Prentice Hall, 1989, ISBN 0-13-336165-9

[2] ANPR web site [Online] http://www.anpr-tutorial.com

[3] Apple documentation reference. [Online] https://developer.apple.com/library/mac/#documentation/AVFoundation/Reference/AVCaptureSession_Class/Reference/Reference.html

[4] Apple Technical Q&A QA1744. [Online] http://developer.apple.com/library/ios/#qa/qa1744/_index.html#//apple_ref/doc/uid/DTS40011134

[5] Călin Enachescu, Calcul Neuronal, Editura Casa Cărții de Știință, 2009, ISBN 978-973-13-460-8

[6] Chris Solomon, Toby Breckon, Fundamentals of Digital Image Processing: A Practical Approach with Examples in MATLAB, Wiley, 2011, ISBN 978-0470844731

[7] De-Shuang Huang, Kang-Hyun Jo, Hong-Hee Lee, Hee-Jun Kang, Vitoantonio Bevilacqua, Emerging Intelligent Computing Technology and Applications. With Aspects of Artificial Intelligence, Springer, 2009, ISBN 978-3642040191

[8] Gary Bradski, Adrian, Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly, 2008, ISBN 978-0596516130

[9] Jesse Hansen, A MATLAB Project in Optical Character Recognition [Online] http://www.ele.uri.edu/~hansenj/projects/ele585/OCR/OCR.pdf

[10] John Canny, A Computational Approach to Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI 8, Issue 6, November 1986
[11] J.R. Parker, Algorithms for Image Processing and Computer Vision, 2nd Edition, Wiley Publishing, 2010, ISBN 978-0-470-64385-3
[12] Mark S. Nixon, Alberto S. Aguado, Feature extraction and image processing, Newnes, 2002, ISBN 9780123725387
[13] Ondrej Martinsky, Algorithmic and Mathematical Principles of Automatic Number Plate Recognition Systems, Brno University of Technology, B. Sc. Thesis, 2007
[14] Roman Sládeček, iPhone Application for Number Plate Recognition, Bachelor's Theses, Brno University of Technology, 2011
[15] Tesseract Homepage [Online]: https://code.google.com/p/tesseract-ocr/