DEGREE PROJECT, IN COMPUTER SCIENCE , SECOND LEVEL
STOCKHOLM, SWEDEN 2015

Visual Vehicle Identification Using Modern Smart Glasses

ANDREAS MALMGREN

KTH ROYAL INSTITUTE OF TECHNOLOGY

SCHOOL OF COMPUTER SCIENCE AND COMMUNICATION (CSC)

KTH ROYAL INSTITUTE OF TECHNOLOGY

Visual Vehicle Identification Using Modern Smart Glasses

Visuell fordonsidentifiering med moderna smarta glasögon

by

Andreas Malmgren

Supervisor: Arvind Kumar
Examiner: Anders Lansner
Partner Company: Scania CV AB

A thesis submitted in partial fulfillment for the degree of Master of Computer Science

in the
School of Computer Science and Communication (CSC)

June 2015

Abstract

Visual Vehicle Identification Using Modern Smart Glasses

In recent years wearable devices have been advancing at a rapid pace and one of the largest growing segments is the smart glass segment. In this thesis the feasibility of today's ARM-based smart glasses are evaluated for automatic license plate recognition (ALPR). The license plate is by far the most prominent visual feature to identify a specific vehicle, and exists on both old and newly produced vehicles. This thesis propose an ALPR system based on a sequence of vertical edge detection, a cascade classifier, vertical and horizontal projection as well as a general purpose optical character recognition library.

The study further concludes that the optimal input resolution for license plate detection using vertical edges is 640x360 pixels and that the license plate need to be at least 20 pixels high or the characters 15 pixels high in order to successfully segment the plate and recognize each character. The separate stages were successfully implemented into a complete ALPR system that achieved 79.5% success rate while processing roughly 3 frames per second when running on a pair of Google Glass.

Sammanfattning

Visuell fordonsidentifiering med moderna smarta glasögon

Under de senaste åren har området wearables avancerat i snabb takt, och ett av de snabbast växande segmenten är smarta glasögon. I denna examensuppsats utvärderas lämpligheten av dagens ARM-baserade smarta glasögon med avseende på automatisk registreringsskyltigenkänning. Registreringsskylten är den i särklass mest framträdande visuella egenskapen som kan användas för att identifiera ett specifikt fordon, och den finns på både gamla och nyproducerade fordon. Detta examensarbete föreslår ett system för automatisk registreringsskyltigenkänning baserat på en följd av vertikal kantdetektering, en kaskad av boostade klassificerare, vertikal och horisontell projektion samt ett optiskt teckenigenkänningsbibliotek.

Studien konstaterar vidare att den optimala upplösningen för registreringsskyltdetektion med hjälp av vertikala kanter på smarta glasögon är 640x360 pixlar och att registreringsskylten måste vara minst 20 pixlar hög eller tecknen 15 pixlar höga för att registreringsskylten framgångsrikt skall kunna segmenteras samt tecken identifieras. De separata stegen implementerades framgångsrikt till ett system för automatisk registreringsskyltigenkänning på ett par Google Glass och lyckades känna igen 79,5% av de testade registreringsskyltarna, med en hastighet av ungefär 3 bilder per sekund.

Acknowledgements

I would like to express my gratitude to my supervisor Arvind Kumar for the useful comments, remarks and engagement through the learning process of this master thesis. Furthermore I would like to thank my supervisor at Scania, Bashar Mengana, for his passion for the project as well as helping hand and insight into Scania. I would also like to extend a thank you to Scania for providing various resource, workspace and the Google Glass used in this thesis. Lastly, I would like to thank my examiner, Anders Lansner, for his remarks and comments for this thesis.

List of Figures

List of Tables

Abbreviations

ALPR	**A**utomatic **L**icense **P**late **R**ecognition
AR	**A**ugmented **R**eality
CCA	**C**onected **C**omponent **A**nalysis
HMM	**H**idden **M**arkov **M**odel
LPE	**L**icense **P**late **E**xtraction
LPCS	**L**icense **P**late **C**haracter **S**egmentation
LPCR	**L**icense **P**late **C**haracter **R**ecognition
MLP	**M**ulti-**l**ayer **P**erceptron
OCR	**O**ptical **C**haracter **R**ecognition
PNN	**P**robabilistic **N**eural **N**etwork
SVM	**S**upport **V**ector **M**achine

Contents

Dedicated to my loving and supporting parents

Chapter 1

Introduction

First the mobile phone revolutionized how we communicate, followed by the release of Apple's iPhone which forever changed how we interact with mobile phones and other devices with embedded screens. In recent years wearable devices have been advancing at a rapid pace. Recently Google introduced Glass, an unobtrusive device that is worn like conventional glasses but with a compact computer, high-resolution camera, microphone, bone-conduction transducer, touchpad and display as well as wireless connectivity built into the frame. This kind of device is more generally known as smart glasses, and Google are not the only company trying to get on the train, as analysts predict that by 2018, more than 25 million head-mounted displays, such as smart glasses, will have been sold [1].

Common for all of the smart glasses are that they are primarily operated by voice commands, head and finger gestures or eye winks, making it possible for the wearer to operate the device while their hands are free to do something else. This, in combination with the conveniently located display and the availability of the forward-facing camera, makes the smart glasses exceptionally well suited devices for augmented reality (AR).

AR is commonly defined as when a view, direct or indirect, of the physical real world is augmented with data. Such data can be comprised of e.g. text and graphics which help the user perform a task, and there are literary an unlimited number of applications where AR can be useful. Imagine a police officer could know if a vehicle is stolen just by looking at it, or a mechanic getting vehicle health information by glancing at the license plate.

In the trucking industry there are multiple opportunities to make use of AR. By law each driver need to perform a daily security check of his/her vehicle before taking it into

use. Since drivers do not use the same truck each day and the procedure is different for each truck type, the inspection is often neglected.

One way of easing the procedure is by implementing automated license plate recognition (ALPR) in a pair of smart glasses and using AR to guide the driver through the correct inspection for the truck type in question. This would reduce time consumption for the inspection as well as make sure the correct checks are performed. Further, many trucks connects to trailers in order to carry goods. A problem is that the driver sometimes misread the license plate of the trailer and accidentally transports the wrong trailer, which in turn result in high overhead costs and delays. By having ALPR in a pair of smart glasses confirm that the license plate corresponds to the correct trailer, such mistakes could be completely eliminated.

As truck drivers often are very conservative with regards to technology it is of great importance that such a device works fast and without delays and with high recognition rate for the truckers to trust and use the system.

1.1 Partner company

This thesis was performed in conjunction with Scania CV. Scania is a major international manufacturer of trucks and busses. The company is active in the premium segment of the truck manufacturing market and are known to be a frontier in new technology. For future projects Scania is very interested in the potential of wearables, and smart glasses in particular. License plate recognition is one of the most robust, backward and forward compatible methods to visually identify a specific vehicle, therefore Scania has an interest in what can be accomplished in the field of ALPR on this type of new platform.

1.2 Problem Formulation

ALPR has many useful applications, but is also a very difficult problem. The majority of current research, therefore, reduces the complexity by establishing some delimitations, such as using stationary capturing devices with high performing and dedicated hardware [2].

When the device is fixed, the system can easily be calibrated to compensate for any inclination and complex backgrounds. Further, in such systems the distance to the vehicle is know, illumination can be artificially improved as well as plate detection can be improved by knowledge of where in the captured image the license plate might appear. In

real world applications using smart glasses, many of these delimitations are not realistic, and most available research are not performed with portable devices in mind [3].

Firstly, as user brings the device with them, all of the above mentioned environmental aspects will vary and be unknown. Secondly, the performance in modern smart glasses are no where near the performance of modern computers and dedicated signal processors, and will most likely not be for years due to the limited capacity of today's batteries. Lastly, as the methods in the literature are tested using Intel or AMD processors, there is no guarantee that they will perform the same on a general-purpose ARM-based system (which most available smart glasses are built upon).

All of these variables makes it a completely new challenge to perform ALPR using modern smart glasses rather than e.g. a stationary dedicated device.

1.3 Purpose

The purpose of this thesis is to investigate how an automatic license plate recognition system should be implemented into a modern pair of ARM-based smart glasses, as well as to determine if it is possible for such a system to achieve high recognition rate while maintaining low execution time.

1.4 Research questions

The main research question of this thesis is: Can an automatic license plate recognition system be implemented into a modern pair of ARM-based smart glasses and achieve high recognition rate while maintaining acceptable execution time?

In order to answer the main research question it has been broken down into three sub-questions:

- What license plate localization and extraction methods are most suitable for modern ARM-based smart glasses?
- What character segmentation and optical character recognition methods are most suitable for modern ARM-based smart glasses?
- What is the best recognition rate that can be achieved using an ALPR system developed for modern ARM-based smart glasses in relation to execution time?

1.5 Delimitations

As previously mentioned, detection of vehicle license plates is a challenge task with many variables. Due to the short time-window of the master thesis the scope of this research will be limited.

Each country have individual license plate configurations. Due to the lack of a large dataset with license plates, this research will be limited to standard Swedish 520x110 millimeter license plates with six black characters, three letters and three numbers, on white background. Further, the research will assume day-time lightning conditions as well as the vehicle being scanned is stationary. As it is fairly unnatural for humans to inspect a vehicle while tilting their head, a maximum roll of ± 10 degrees will be allowed.

The research will also be limited to smart glasses running the Android operating system using an ARM-processor, which most available smart glasses today do.

1.6 Report layout

Chapter 2 studies the methods and results of previous research in the separate stages of ALPR and also provides a general background to what aspects of ALPR makes it hard to perform. In Chapter 3 the theoretical framework needed to understand the methods used in this thesis are explained and the various libraries used are introduced. Chapter 4 explains decisions and in detail describes the separate parts of the study and what data each part is intended to collect. Chapter 5 objectively presents the collected results from each of the experiments. The results presented in the previous mentioned chapter is then discussed and analyzed in Chapter 6 and ultimately leads to Chapter 7 where the findings are concluded and the research question is answered in a compact format as well as future research is suggested.

Chapter 2

Previous research

2.1 Automatic License Plate Recognition

Automatic license plate recognition is used for autonomous vehicle identification and has multiple applications, such as automatic parking attendant [4], toll collection [5], speed limit enforcement [6], identification of stolen cars [4] and driver support [7] [2]. The main goal is to translate a license plate shown in an image or a video stream to digitized text.

It is a complex task which is made even more difficult as the system needs to consider multiple plate variations [8] [9] [10] [11]:

1. The placement where a plate can be found in an image may vary
2. The different sizes a plate can have due to factors such as the distance from the camera to the vehicle or camera zoom. Further each country may have their own size and dimension specifications for the license plate
3. The image may contain multiple or no plates at all
4. The plate background or characters may have different colors or the camera used may capture incorrect colors
5. The font used in different countries may vary
6. Custom license plate text
7. The image may be captured at an angle
8. The plate may be completely or partially covered by dirt

9. Frames or screws that may disturb recognition

Further, the ALPR system will have to accommodate certain environmental variations such as [8] [9] [10]:

1. The illumination variations caused by vehicle lights, weather conditions and time of day
2. Conditions such as the background containing plate-similar patterns, numbers or text or other disturbing information
3. Physical damage to the plate

In order to handle these variations many commercially available products rely on specialized hardware, such as infrared cameras and dedicated processing units, and for many applications the workplace is design with optimized lightning. All these variables show an extremely complicated problem, but using proper computer image processing techniques, many of these difficulties can be controlled [8].

The typical ALPR system structure is based on four distinct stage. First an image containing the vehicle and its visible license plate is acquired by a digital camera. The second stage locates the coordinates of the license plate and extracts that region, resulting in a smaller image containing with just the license plate. The third task is to isolate each of the individual characters and numbers and pass them separately to the last module which performs optical character recognition (OCR) on each of the segments in order to identify the character [7] [12] [13].

2.2 License Plate Extraction

License plate extraction (LPE) is the actual task of detecting, locating and extracting the license plate from within a larger image, as the plate can exist anywhere in the image. The input is usually an image of a vehicle with some background and after LPE is performed the output should only contain the license plate in question. Errors in the output from LPE greatly reduce performance of ALPR systems. The literature suggests a wide variety of approaches with varying performance. As often more than half of the ALPR execution time is spent with localizing and extracting the license plate [14], it is a module that is very important to select optimal methods for.

Hongliang and Changping [15] use a mathematical morphology operation called "top-hat" for a highway ticketing system implementation. The images were captured at a

fixed angle and a known distance as the morphological operations relate to the object dimensions. The implementation achieved 97% successfully recognised plates as well as 22 false positives from a test sample of 105 car images with limited disturbing background. Hongliang and Changping also comments on the implementation being "a bit slow".

Miyamoto et al. [16] achieve "remarkably high accuracy", more than 99%, using a 2D pattern search trying to find numerics and numeric-like figure. When some candidates are found the most promising is judged by its geometrical shape and the layout pattern of the numerals in the plate. According to the authors they obtain high performance independent of the positions, features and configurations of the license plates, but the approach is time consuming and have a computational time complexity of $O(n^4)$ for images of size $n \times n$ [17].

Using connected component analysis (CCA) Zheng, Zhao and Wang [18] achieved near 100% plate location rate while benchmarking and outperforming three earlier proposed algorithms; 'line sensitive filters [19], 'row-wise and column-wise DFTs [20] and edge image improvement [21]. The test was performed on a Pentium 4, 2.4 GHz PC with 256 MB RAM with an average execution time of 47.9 ms. Mahini and Dorri [22] use larger images in a similar approach with vertical edges, morphological operation, and color analysis and manage to achieve 96.5% detection rate with an execution time of 300 ms on a Pentium 4 PC with 512 MB RAM.

Another approach to LPE is detecting edges in the inputs image and then using Hough transformations to find the license plate region. An advantage with this algorithm is that is allows for up to 30 degree inclination when detecting straight lines [23]. Unfortunately the Hough transformation is computationally heavy and also consumes a lot of memory [24] [25]. Therefore, Duan et al. [24] suggested a combination of Hough transformation and contour algorithm in order to produce higher speed and increased accuracy of 98,76% plate recognition rate. The test was performed on a Pentium 4, 1.4 GHz PC with 512 MB RAM with an average execution time of 0.65 seconds per image.

Cascade classifiers for LPE have in recent years grown in popularity. Wang and Lee [10] used a cascade framework with Haar-like features on plates of different sizes, formats and illumination with a detection rate of over 99% when executed on a Pentium 4, 3.0 GHz PC achieving 38 processed frames per second. The use of Haar-like features allows for the implementation to be relatively invariant to changes in brightness, size (distance), color and position of license plates in the processed image, however, it can also be sensitive to inclination. In [26] the authors propose the use of histogram of oriented gradients to estimate the likelihood of candidates from a Haar-like feature classifier.

Huang et al. [3] perform preprocessing of the input image and then search for the license plate using histogram projections along the x-axis to determine the height of the plate and horizontal segmentation. Using a window based on the license plate dimensions a search for the final plate is performed along the horizontal segment.

Ref.	**Main method**	**Image res.**	**Exec. time**	**Success rate**
[15]	Mathematical morphology on binary image	-	-	97%
[18]	Vertical edge extraction using Sobel operator	384x288 pixel	47.9 ms	$\backsim$ 100%
[22]	Vertical edge extraction, morphological operations and color analysis	800x600 pixel	300 ms	96.5%
[24]	Hough transformation and contour algorithm	800x600 pixel	650 ms	98.76%
[10]	Haar-like feature-based cascade classifier	640x480 pixel	26 ms	99.86%
[3]	Histogram projection with search window on binary image	320x240 pixel	293 ms (including LPCS and OCR)	96.7%

TABLE 2.1: The success rate, execution time and image input resolution for previous research on license plate extraction.

2.2.1 Preprocessing

The output image from the LPE might still contain imperfections, depending on the LPE module. Such problems can be nonuniform brightness, tilt and inclination [24] and often need to be corrected in a preprocessing step before license plate character recognition can be performed with good results.

2.2.1.1 Vertical and horizontal tilt correction

Xu et al. [27] use vertical and horizontal projection analysis in combination with the knowledge of the license plate size to find four vertexes. These are then used to perform the geometric transformation and retrieve the upright rectangle image. Pan, Yan and Xiao [28] propose a least square method to correct vertical and horizontal tilt in license plate images. Pan, Xiong and Yan [29] propose a new method for correcting vehicle license plate tilt, but seem to experience a loss in character quality.

2.2.1.2 Thresholding

As the lightning differences in separate parts of a scene might vary greatly, global thresholding is often not an appropriate technique. Therefore local or adaptive thresholding is more common in the literature. Llorens et al. [11] suggest a method where the threshold is calculated from the mean gray level in a $n \times n$ window centered over the pixel. Zhang and Zhang [30] instead propose an object enhancement algorithm based on the assumption that the characters in a license plate image accounts for 20% of the pixels. So theses pixels are enhanced while the remaining pixels are weakened. Ying et al. [31] use the Bernsen algorithm with good performance. Coetzee, Botha and Weber [32] achieve good results using the Niblack algorithm which calculates a local binarization threshold using the local standard deviation and mean in a 15×15 neighbourhood. Huang et al. [3] apply Otsu's method [33] to dynamically determine the threshold for each region. This lowers the impact of environmental noise in the input image. [22] and [34] also exploit the Otsu threshold algorithm to binarize their plate candidates.

2.3 License Plate Character Segmentation

The objective of the license plate character segmentation (LPCS) is to partition the isolated license plate image into segments containing the individual letters and numbers [35]. The problem is challenging partially due to noise in the image, illumination differences and shadows or other artifacts present in real world scenarios. Without correctly segmented and extracted characters the performance of the OCR will be reduced even for the most robust algorithms available. An extracted license plate may contain two types of characters, distinct and indistinct characters. Distinct characters are letters and numbers on the plate that are clearly separated while indistinct characters are connected to each other [10].

The most simple and common segmentation procedure in the literature [28] is based around the brightness difference of the characters and background. This method works well for distinct characters, but will not work if the characters are inseparable. It will also suffer reduced performance if the analyzed image contains noise. This method is used where vertical and horizontal projections of the binary license plate image are performed to obtain a vector with the number of white pixels for each vertical and horizontal line in the image [3] [24] [36] [37] [38] [39] [40]. The minimum values of the vectors allows the image to be segmented into characters individual.

CCA has been proposed [41], [22], [10], in combination with object measurements such as area, hight, width and orientation, for character segmentation. CCA labels 8-neighbour

clusters of pixels into components and then using the knowledge about the characters to match the components to potential character segments. This method is simple and straightforward. However, even though it can handle some rotation, it fails when there are indistinct characters.

Both of theses methods fail if the plate image is severely degraded or contain indistinct characters. Nomura et al. [35] argues that a mathematical morphology approach is more suitable for segmenting such complex plates. Using information about the maximum number of letters and numbers in combination with natural segmentation points and mathematical operations to adaptively determine the optimal distribution of segmentation lines.

Capar and Gokmen [42] suggest the use of a statistical boundary shape model with an implicit contour based segmentation method to allow the characters to be segmented and recognized in the same module. The authors state that similar previously proposed systems have required high computational power, but at the same time mention that their system has one less optimization step and should be faster. Unfortunately they never explicitly show their performance increase in the paper. Kim, Jang and Kim [43] also use contour tracking to perform character segmentation. Since both methods utilize the characters contours, their performance are greatly decreased if indistinct characters are present.

Franc and Hlavac [44] propose a method based on machine learning for use with noisy low resolution images. A hidden Markov chain is used in conjunction with specific knowledge about the license plate, such as the number of segments needed and that the plate characters can be segmented with equal (but unknown) width. The proposed method achieved a success rate of 96.7% using real-world data.

2.4 License Plate Character Recognition

The extracted and isolated characters from the LPCS need to be recognized in order to produce the digital license plate number as an output. The process of converting text from images is often referred to as optical character recognition (OCR) and can be performed using a wide variety of methods. But as the input data still might suffer from issues such as fluctuations in character thickness, partial cut-offs, noise, etc [16], theses must be taken in to account or corrected in order to achieve optimal results. Depending on the severity it might be enough to e.g. resize the characters to one standardized size before recognition.

Most OCR methods are based on one of two types of data; raw image data or extracted features. The first being the untreated pixel values in the input image while the latter being based around the concept that some pixels are more important in distinguishing the character than others. Feature extraction can also be used to reduce the size of data that is too large to be processed.

2.4.1 Using Raw Pixel Data

Template matching is a simple and straightforward approach to using raw pixel data [7]. Most methods that utilize template matching are performed using binary images, as lightning conditions produce too large variety in e.g. gray-scale sources. The binary character segment is then compared to templates and their similarity are measured, the character of the template with highest similarity is then chosen as candidate. What often differ between the studies using template matching is the similarity measuring techniques. Template matching works well for single-font, fixed size and non-rotated scenarios [28].

Sarfraz, Ahmed and Ghazi [7] use normalized inputs of 40×40 pixel images and matched against the template database using what seem to be the simple Hamming distance. This approach was tested on a large image database producing 95.24% accuracy. Lee, Kim and Kim [45] normalize their images to 40×40 pixels, but instead calculate the Jaccard similarity coefficient in order to classify with 98.8% recognition rate and average execution time per character of 286 ms (incl. LPCS) on a 50 MHz 486 PC. Miyamoto et al. [16] apply a similar approach but instead use Mahalanobis distance to compare similarity and also implement Bayse decision theory biased to the safe side. No partial results are presented for the OCR, but the system achieve 99% recognition rate for all three ALPR steps. In [46] the authors try performing OCR through normalized cross-correlation, but only attain 92.7% recognition rate using 275 images. Using 11 templates for each character, each with different rotation, Natio et al. [34] obtained 97.3% recognition rate for plates with up to 40 degree rotation.

2.4.2 Using Extracted Features

2.4.2.1 Feature Vectors

As each pixel in an image does not provide equal amount of information towards recognizing a character, one could try to reduce the whole image into a set of features which describes the specific character. While it reduces the processing time it also, if chosen

correctly, increases performance for distorted characters compared to template matching [28]. The extracted features are often stored in a matrix called feature vector.

In [47], the authors build the feature vector from blocks of 3×3 pixels in a binary character image. The number of black pixels in each block is then counted and used as features. [24] use slightly overlapping windows of 9×9 pixels and scan the image from left to right and top to bottom, calculating the ratio of foreground pixels in the window. There values are then used as features.

Ko and Kim [48] suggest an approach to feature extraction where the character is scanned along a vertical and horizontal line. While scanning, the number of transitions between character and background is counted and used as feature vector. This approach is robust to rotation as the same feature vector is produced independent of rotation.

Gabor filter is used for building feature vectors in [49]. Abdullah, Khalid and Yusof [50] propose the use of Kirsch edge detector for feature extraction but concludes that it might be insufficient for feature extraction.

The authors in [11] extract a feature vector by dividing the input image into a grid of $20 \times N$ units, where N is proportional to the width of the license plate image. In each cell the horizontal and vertical derivatives are computed as well as the average gray level and those are used as features.

[41] use four discrete-time cellular neural networks to generate features based on vertical and horizontal projection, as well as vertical and horizontal connected component count. Each of theses features were then transformed into five inputs in a feature vector.

2.4.2.2 Classifiers

The extracted features are used to classify the character segments into letters and numbers. There are many types of classifiers, but the issue with many of them are that they need extensive data sets to be trained with before they can perform classification.

Llorens et al. [11] suggest the use of hidden Markov models (HMM). They train one HMM for each character and achieved 98.1% accuracy rate. The authors states that most errors were due to blurry images and overexposure. Duan et al. [24] also make use of a HMM and trains it with 60 images of each class, producing 97.52% recognition rate.

The authors in [14] implement four separate support vector machines (SVMs) for classification of characters for each distinct character region in korean license plates. In the case of numbers they utilize one SVM for each number respectively. Doing so they

obtained a recognition rate of 97.2% when tested on 400 video clips containing 10 frames each.

A multi-layered perceptron (MLP) is a feedforward artificial neural network model that is commonly used in the literature for character identification. Many of them are implemented in a similar fashion where they are trained using error backpropagation, the drawback of this technique is that it requires many training cycles and large quantities of training data in order to perform well. Nijhuis et al. [41] suggest a MLP with 24 input-, 15 hidden- and 36 output neurons (one for each character) and by limiting the recognition to be classified as successful only if the value of the output neuron exceeds 0.85 and all other output neurons having values lower than 0.25, they achieve a recognition rate of 98.51%.

Another type of neural networks, probabilistic neural networks (PNN), are also described in the literature because of robustness and superior training time compared to feedforward backpropagation networks [51]. Özürk and Özen [52] apply PNNs to character recognition and produce a recognition rate of 96.5% with low execution time. Hu, Zhu and Zhang [51] also make use of PNNs in their implementation and obtain an impressive 99.5% recognition rate, significantly outperforming template matching using the same test images. The performance difference is even larger when severely noisy images are used.

Some of the researchers have used general-purpose optical character recognition libraries. Zheng and He [40] implemented the Tesseract OCR library as their character recognition module, and achieve an incredible performance of 98.7% recognition rate.

Ref.	**Main method**	**Exec. time**	**Recognition rate**
[7]	Template matching with Hamming distance	-	95.24%
[45]	Template matching with Jaccard similarity coefficient	286 ms	98.8%
[16]	Template matching with Mahalanobis distance	-	99% (complete ALPR system)
[46]	Template matching with normalized cross-correlation	-	92.7%
[34]	Template matching with multiple character templates	-	97.3%
[11]	One HMM trained for each character	-	98.1%
[24]	One HMM trained for all characters	-	97.52%
[14]	One SVM for each character group, separate SVMs for each number 0-9	-	97.2%
[41]	MLP with 24-15-36 configuration and output criterion	-	98.51%
[52]	PNN	-	96.5%
[51]	PNN	-	99.5%
[40]	Tesseract OCR libarary	-	98.7%

TABLE 2.2: The recognition rate and execution time (if available) for previous research

Chapter 3

Theoretical framework

3.1 Swedish License Plates

There are many color and shape configurations of Swedish license plates depending on the vehicle and situation, such as diplomatic and taxi vehicles. In this thesis the focus will be on the most common license plate, which has a single row of six characters with white background and black characters. All new license plates also have a EU-symbol with blue background located on the left side of the plate, see Figure 3.1.

Since 1973 the format of three letters followed by three digits have been used. Even though the character configuration have been constant, the swedish department of motor vehicles (currently known as Transportstyrelsen) have changed the typeface multiple times. In the last 20 years a customized Helvetica font, a font face similar to Alte DIN 1451 Mittelschrift [53] and now a custom font called Swedreg have been used.

FIGURE 3.1: An image of the standard 520x110 mm Swedish license plate as of 2014.

3.2 Algorithms and Methods

3.2.1 Smoothing

Smoothing, or blurring as it is also called, is a frequently used image processing technique to reduce noise in an image. The most common approach to smoothing is by applying a linear filter. In this thesis a simple 5x5 normalized box filter will be used. Each output

pixel is determined as the mean of it kernel neighbours. The kernel $\mathbf{K}$ is defined as in 3.1.

$$\mathbf{K} = \frac{1}{\mathbf{K}_{width} \cdot \mathbf{K}_{height}} \begin{bmatrix} 1 & 1 & 1 & ... & 1 \\ 1 & 1 & 1 & ... & 1 \\ . & . & . & ... & 1 \\ . & . & . & .. & 1 \\ 1 & 1 & 1 & .. & 1 \end{bmatrix} \tag{3.1}$$

3.2.2 Binarization

In regular, global, thresholding an arbitrary threshold value is decided on before the thresholding is performed. In this case it is impossible to know beforehand if the value chosen is good when applied to a previously unseen image. Otsu's binarization consider an image with two histogram peaks, bimodal image, and tries to find the threshold value so that the weighted within-class variance is minimized in 3.2.

$$\sigma_w^2(t) = \ q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \tag{3.2}$$

$$q_1(t) = \sum_{i=1}^{t} P(i) \quad \& \quad q_2(t) = \sum_{i=t+1}^{I} P(i) \tag{3.3}$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \quad \& \quad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)} \tag{3.4}$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \ \frac{P(i)}{q_1(t)} \quad \& \quad \sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \ \frac{P(i)}{q_2(t)} \tag{3.5}$$

where

Symbol	Explanation
t	Threshold level
$P(i)$	Pixels with intensity value i
$\sigma_w^2(t)$	Within-class variance
$q_x(t)$	Number of pixels in each class x
$\mu_x(t)$	Mean of each class x
$\sigma_x^2(t)$	Variance for each class x

TABLE 3.1: Variable explanation for the Otsu algorithm.

3.2.3 Morphological Closing

Morphological closing of a binary image is obtained through performing morphological dilation followed by erosion. In this thesis it will be used to close small holes in the binary image during license plate candidate detection.

The dilation process slides a structuring element over the image for probing and expanding the foreground shapes. While sliding the structuring element, each location where the element covers any part of the foreground shape, the center pixel of the element will be given the value "1", otherwise "0", allowing the shape to expand. The process of erosion is similar, but here only locations where the complete structuring element is inside the foreground shape will be given the value "1". The process is illustrated in Figure 3.2.

FIGURE 3.2: An illustration of the morphological closing. From left to right: original shape, erosion with a round element, result of the erosion, dilation of the eroded shape, result of the dilation.

3.2.4 Vertical Edge Detection

There are various ways to detect edges in a digital image. An edge can be explained as when the pixel intensity drastically changes from one pixel to a neighbouring pixel. Using derivatives is a good way to express these changes as an edge location would appear as a local maximum or minimum. The Sobel operator can be used to compute an approximation of the gradient of a digital image intensity function.

Assuming that the input image is $\mathbf{I}$, we would calculate the vertical gradient $\mathbf{G}_x$ at each point by convolving $\mathbf{I}$ with a 3x3 kernel as shown in 3.6.

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{I} \tag{3.6}$$

3.2.5 Haar-like features

Features in machine learning aim to encode unique information about a certain object or image. Their main purpose is to replace raw pixel values as input features to a machine

learning algorithm, in order to reduce the within-class variability, resulting in a more robust classifier. Viola et al. first introduced Haar-like features for face detection, but they have become popular for other types of object detection, primarily because of its speed [54]. The Haar-like features of any size can be calculated in constant time, $O(1)$, with the use of integral images.

The integral image is a data structure with the same dimensions as the analyzed image and where each element's value, I, is the sum of all pixel values, i, above and to the left of itself, as in 3.7 [55]. The integral image is efficiently computed in a single pass over the image as each element in the integral can be calculated using 3.8. Once the integral image is obtained, any rectangle in the image can be evaluated in constant time using four values as in 3.9. The integral image is visualized in Figure 3.3.

$$I(x,y) = \sum_{\substack{x' \leq x \\ y' \leq y}} i(x',y') \tag{3.7}$$

$$I(x,y) = i(x,y) + I(x-1,y) + I(x,y-1) - I(x-1,y-1) \tag{3.8}$$

$$\sum_{\substack{x_0 < x \leq x_1 \\ y_0 < y \leq y_1}} i(x,y) = I(D) + I(A) - I(B) - I(C) \tag{3.9}$$

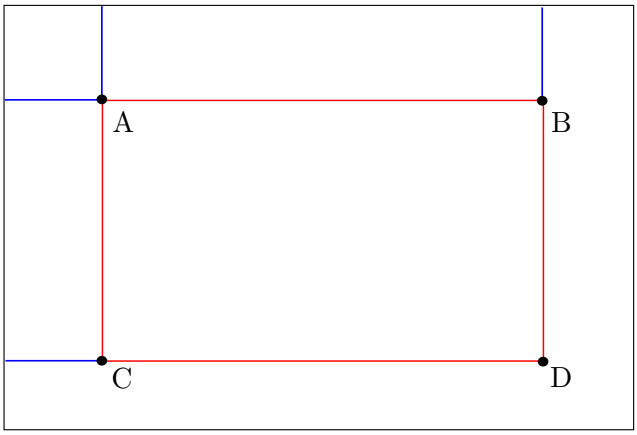

FIGURE 3.3: A visualization of the points used from the integral image to calculate a sub-area.

The standard Haar-like features presented by Viola et al. consists of a class of local features that are extracted by subtracting the pixel value sum of a rectangular sub-region of the feature from the remaining region. As the regions are rectangular, the features are easily calculated using the integral image. Lienhart and Maydt extended the set of Haar-like features by rotating them 45 degrees, as can be seen in Figure 3.4 [56]. By rotating the integral image as well, the extended features can still be efficiently calculated.

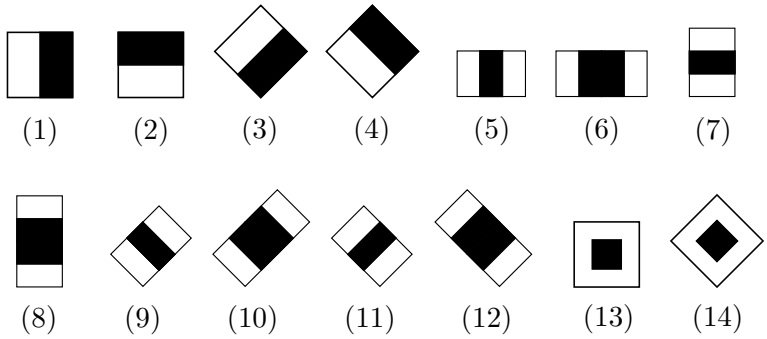

FIGURE 3.4: An illustration of straight and rotated Haar-like features. The white areas are subtracted from the black areas.

3.2.6 Classifier

3.2.6.1 Boosting

Boosting is a powerful learning concept which in this thesis is used as the basic classifier. It is based around the fact that many "weak" classifiers with low computational cost can, when combined, perform as a powerful "strong" classifier. The only criterion required of a weak classifier is that it performs better than chance, less than 50% error over any distribution, which allows it to be simple and fast. The final classification is then based on a weighted vote of the weak classifiers, as in 3.10.

$$\underbrace{F(x)}_{Strong} = \underbrace{\alpha_1 f_1(x)}_{Weak} + \alpha_2 f_2(x) + \alpha_3 f_3(x) + ... \tag{3.10}$$

3.2.6.2 Cascade of Classifiers

To perform classification using a boosted classifier with many features is time consuming. A faster way is to use a cascade of classifiers with only a few features instead. The cascade is a series of classifiers (rejecters) where each rejecter detects almost all objects of interest but rejects a large number of non-objects. Each stage is increasingly more computationally expensive, but by discarding most of the non-objects early in the cascade saves lots of computations, as visualized in Figure 3.5.

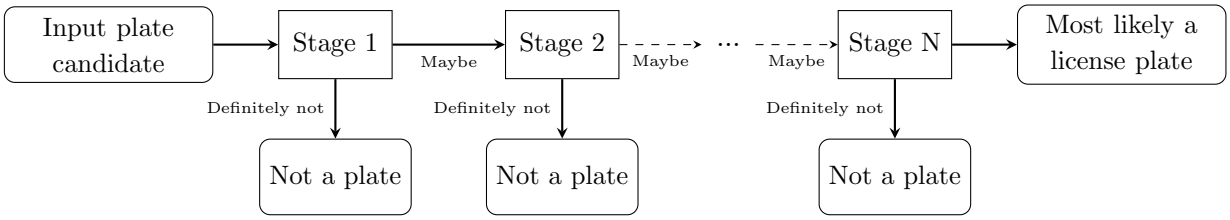

FIGURE 3.5: A visualization of the flow through a cascade classifier.

3.2.6.3 Training

During the supervised training the adaptive boosting method AdaBoost is used to select which few features should be applied in each of the classifier stages. There are multiple versions of AdaBoost, but for this paper we use Gentle AdaBoost [54][57].

At each round of boosting, the feature-based weak classifier is added that has the lowest error in classifying the training samples. The iteration proceeds until a set of defined criteria are met, such as the minimum acceptable hit rate and the maximum acceptable false alarm rate. For each increasing stage in the classifier cascade, the number of weak classifiers needed to achieve the defined hit rate and false alarm rate is higher.

3.2.7 Classifying

When the classifier has been trained, it can be used to classify a region of interest with the same size as the training data. The classifier output is a "1" if a license plate is detected, otherwise a "0". To search the whole input image for a region of interest, a sliding search window is used to evaluate each possible location, as illustrated in Figure 3.6. To deal with scaling of the object the search window and features are scaled by a scaling factor each time the search window finishes scanning the image, this proceeds until the scaled search window reaches a defined maximum size.

FIGURE 3.6: An illustration of how the sliding search window tests all possible locations for a license plate.

3.2.8 Vertical projection

By projecting the white pixels of a binary image along its rows generate a histogram reflecting the amount of white pixels in each row. The number of white pixels H in a row y of image I can be calculated as in 3.11. Figure 3.7 presents a visualization of the vertical projection.

$$H(y) = \sum_{x=1}^{x_{max}} I(x, y) \tag{3.11}$$

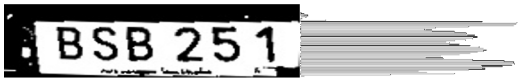

FIGURE 3.7: A visualization of vertical projection on an extracted license plate segment.

3.2.9 Horizontal projection

Horizontal projection is similar to the vertical projection, but instead projects the white pixels along the image columns. Thus the sum of white pixels H in a column x of image I can be calculated as in 3.12. Figure 3.8 presents a visualization of the horizontal projection.

$$H(x) = \sum_{y=1}^{y_{max}} I(x, y) \tag{3.12}$$

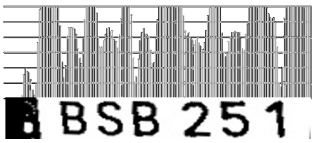

FIGURE 3.8: A visualization of horizontal projection on an extracted and vertically cropped license plate segment.

3.3 Libraries

3.3.1 OpenCV

Open source computer vision (OpenCV) [58] is an industry standard and open source image processing library that is free to use for both commercial and academic use [59]. The library contains more than 2500 optimized algorithms which are accessible through C++, C, Python, Java and MATLAB interfaces on multiple platforms.

3.3.2 Tesseract OCR engine

The Tesseract OCR engine [60] was originally developed by HP, but was released as open source code in late 2005. Since then, there have been extensive contributions to the library by major actors, such as Google, and it has been proven to perform in parity with major commercial OCR applications [61].

Tesseract comes pre-trained for a number of common typefaces, but the developer can train the library for any custom typeface using the integrated and partially automated training procedure [62].

Chapter 4

Methods

4.1 System overview

As is illustrated in Figure 4.1 the proposed system is built on three major stages with individual tasks; extraction, segmentation and recognition. The flow through each of the stages are illustrated in Figure 4.2, Figure 4.3 and Figure 4.4.

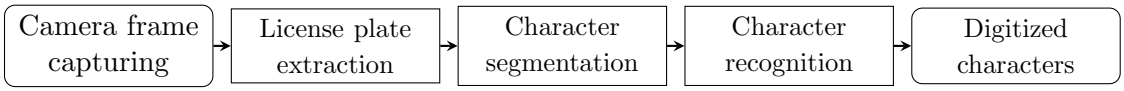

FIGURE 4.1: An overview of the stages in the proposed ALPR system.

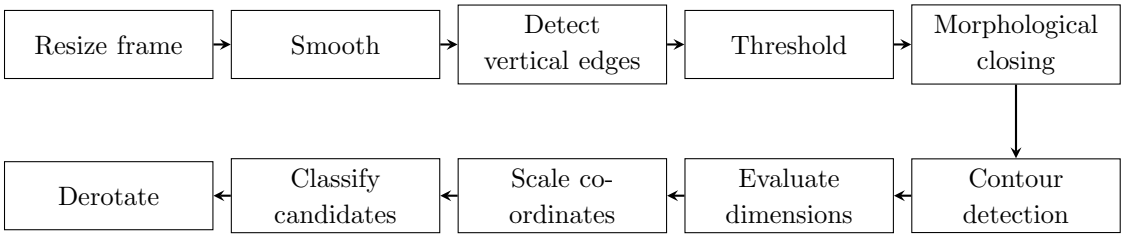

FIGURE 4.2: An overview of the flow through the license plate extraction stage.

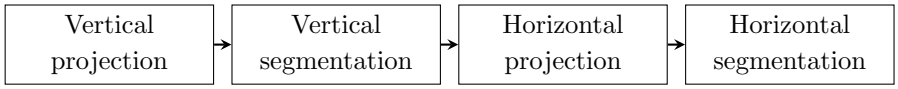

FIGURE 4.3: An overview of the flow through the license plate character segmentation stage.

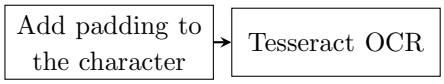

FIGURE 4.4: An overview of the flow through the license plate character recognition stage.

4.2 Stage Optimization

There are multiple factors that play a role in achieved execution time and success rate for each ALPR-stage, and for low powered devices such as the ARM-based smart glasses, keeping the execution time down is a challenge. As there are trade-offs between execution time and success rate we need to find the configuration that produces the optimal success rate to execution time ratio. This was done through a set of three individual configuration benchmarks.

4.2.1 Environment

4.2.1.1 Hardware specifications

FIGURE 4.5: An image of the Google Glass.

Each of the following optimization tests were performed on the Google Glass smart glasses. The glasses use a central processing unit (CPU) from OMAP running at a clock speed of 1.0 GHz, which includes a general-purpose dual-core ARM Cortex-A9 architecture processor. The prism projector located slightly above the wearer's field of vision produces a 640x360 pixel display, which to the wearer looks equivalent of a 25 inch screen from 2.4 m away. The camera is fixed to the frame of the glasses and is therefore always pointing in the same direction as the wearer's head. Even though the camera is only rated for 720p video, it is able to capture a continuous stream of preview frames at a 1080p resolution. The Google Glass operating system is built on top of the Android platform, thus most of the Android application programming interfaces (APIs) are available in the glasses, as well as some APIs developed specifically for the Google Glass.

Specification	Configuration
Operating system	Android 4.4 Kitkat
CPU	OMAP 4430 1.0 GHz
RAM	1 GB
Display	640x360 pixels prism projector
Camera	5 megapixel photos, 720p video at 30 FPS

TABLE 4.1: A summary of Google Glass hardware and operating system specifications.

4.2.1.2 Libraries

During this study multiple libraries have been utilized to speed up development and to ensure that critical functionality is optimized. The use of libraries also increase the generalizabilty and repeatability of the study.

The OpenCV4Android SDK 2.4.10 distribution was used to enable development on the Android platform using Java. Further the Tesseract fork tess-two was used to implement the Tesseract functionally on the Google Glass platform.

4.2.2 License Plate Candidate Detection

The license plate extraction is the most time consuming task in an ALPR system. This is mainly due to the need of a high resolution input image in order to distinguish the license plate from a distance. The wide-angle camera fitted on most smart glasses add to this issue as the captured image contains large areas of background even at close distances. To minimize the number of pixels that the classifier need to process, the first step generates a number of smaller potential license plate candidates from the image.

As license plates contain a high concentration of vertical gradients compared to most backgrounds and the rest of the vehicle's front [63], they can be used to find potential candidates. Before detection of vertical edges, using a Sobel operator, the input image was smoothed and after the resulting grayscale image was reduced to binary format in order to perform morphological closing. To filter out some of the false candidates the contour finding algorithm suggested by Suzoki et al. [64] was used to find the blob contours and only candidates with license plate dimensions were kept.

The performance of the vertical edge detection is deeply dependent on the resolution of the input frame. If the resolution is too low details are lost, but if it is too high the computations take unnecessary long time. Using the optimal input resolution is key to a high performing system. By alternating between five input resolutions (1280x960, 960x720, 800x600, 640x360 and 320x180) together with the configurations for the classifier, the performance was measured in comparison with the execution time.

4.2.3 License Plate Candidate Classification

The license plate candidate detection produces some false positives, therefore a classifier is needed to reject the candidates that does not contain a license plate. A Haar-like feature cascading classifier was used for this task, as it can discard a large number of non-license plate regions using simple and fast computations, while still allowing for some rotation and size variations as well as illumination difference.

4.2.3.1 Dataset

In order to train the classifier a large number of positive and negative images (images that contain license plates and images that does not) were needed. 500 images of cars with Swedish license plates were manually captured in a large parking lot, using a 8 megapixel smartphone camera. Each license plate was using the Swedreg typeface or the slightly older Alte DIN 1451 Mittelschrift based typeface. The images were collected over a period of three days and under varying lightning conditions. The use of a camera other than the Google Glass does not matter in this case, since the small variations that might occur would not be present in the low resolution images as used during the training described below.

Thereafter the plates were manually extracted into compact plate regions containing as little non-plate content as possible. Each plate was converted to grayscale format and normalized to a size of 80x17 pixels. To further improve robustness of the classifier four more samples were generated from each of the extracted regions by randomly rotating it a maximum of ± 10 degrees around the horizontal axis and applying intensity deviation of up to 40 pixel values. This resulted in a total of 2500 samples, 2000 for training and 500 for testing.

The negatives samples were collected from arbitrary images from multiple sources, such as Flickr [65], the UC Irvine Machine Learning Repository [66] and snapshots extracted from videos recorded with a pair of Google Glass. All samples were thoroughly examined to make sure that no license plates were present in the images. The negative samples were cropped and scaled to match the positive sample size of 80x70 pixels. The total number of negative samples collected was 7000.

4.2.3.2 Training

Since the smart glasses provide very limited computing power, offline training was selected to train the model outside of the smart glasses. The classifier used was a cascade

FIGURE 4.6: Examples of positive (left) and negative (right) samples.

of boosted classifiers, working with extended Haar-like features [67]. There are multiple types of boosting algorithms, AdaBoost has proven good performance for multiple types of object detection [54] and Gentle AdaBoost in particular have been argued to be less prone to overfitting and thus provide better generalization performance [67][68]. Therefore the cascade was trained using Gentle AdaBoost with Haar-like features as weak classifiers. The training procedure was configured to produce strong classifiers with the following performance requirements:

Requirement	
Number of stages	16
Minimal true positive rate for each stage	99.9%
Maximal false positive rate for each stage	50%

TABLE 4.2: The requirements for the cascade classifier training.

This configuration allows for a theoretical true positive rate of 98.5% and a false positive rate of 30.5 ppm. The training procedure was very time consuming and took just under three days to finnish using a modern Intel Core i7 3.6 GHz with 16 GB RAM. The output from the training was a XML file with information about each of the classifier's trained stages which could be transferred for use on the smart glasses.

4.2.3.3 Classification

The classifier need to identify whether a plate candidate detected by the vertical edge procedure actually is a license plate or not. To do this a search window was moved across the plate candidate area and every position were checked using the classifier, if no license plate was found then the search window was scaled to a larger size and the search was reinitiated. The smaller the size increments are and the larger size variation we allow for the license plate, the longer detection time we will have. Therefore we need to find which configuration achieves the highest detection rate in relation to cost.

To find the optimal configuration, each of the combinations were tested with plate candidates selected by the vertical edge procedure. This was done by varying the search window scale factor between 3% and 13%. The candidates themselves came from 125

captured images of truck fronts from varying natural observation distances (ranging from 1.5 to 5 meters). Execution time and whether a plate was detected was captured and the average result for each configuration was calculated. A plate was considered detected if the classifier produced only one true positive and no false positives. Keeping low execution time is important, therefore a simple ratio between the success rate and the execution time for each configuration was calculated, in order to compare the success of each configuration with regard to execution time. This ratio might promote really fast but low-performing configurations, as 1% success rate at 10 ms would produce a high ratio. To ensure that this was not the case, the ratio was also manually checked to be reasonable.

To be able to classify a license plate from 5 meters distance, the full resolution of the Google Glass preview frame need to be utilized in order for the plate to be large enough for the classifier to find. Therefore, the license plate candidate coordinates passed from the candidate detection step needs to be scaled to the full 1920x1080 pixels resolution and then the original camera frame is used to crop the license plate candidate to be used with the classifier.

4.2.4 License Plate Character Segmentation

The segmentation method, vertical and horizontal projections, was chosen for its simplicity and the fact that many researches have used it with good results. The segmentation benchmark was performed to determine the optimal configuration with regards to success rate versus execution time. It is also important to establish the minimum required license plate resolution needed to successfully segment the characters.

By varying the height of the license plates feed to the segmentation module between 15 pixels and 35 pixels, the success rate and execution time could be recorded. Each of the plate heights were tested on 100 separate derotated and cropped license plates while the performance was recorded. A license plate was considered successfully segmented only if all 6 characters were segmented into tightly fitted boxes and without any false positives.

4.2.5 License Plate Character Recognition

The popular general purpose and open source OCR library Tesseract [60] was selected because of its success in other applications [69] as well as previous research on ALPR [40].

Since Transportstyrelsen does not have the typefaces of the Swedish license plates in an easily available digital format, Tesseract was trained to recognize the typeface Alte

DIN 1451 Mittelschrift, which is similar to the most common typeface in Swedish license plates.

It is critical for Tesseract to get as high quality input as possible. But we need to determine if there is a significant execution time trade off for using higher resolution input and if there is a minimum input resolution for which we still can achieve decent recognition rates.

To test this, 296 letters and 308 digits were extracted and labeled from real-world license plate images. The characters were feed to the trained Tesseract library with varying character heights, from 10 pixels to 25 pixels, and the execution time as well as recognition rate were recorded. A character was considered correctly recognized only if the module output the correct digitized character.

4.3 Feasibility Study

All three stages; extraction, segmentation and recognition were combined and implemented into a complete ALPR application for the Google Glass. The same environment as described in Section 4.2.1 was used. The extraction stage was configured using the settings found most promising in Section 4.2 and the input images were not allowed to be smaller than the lowest input height found in Section 4.2.4. A fast probabilistic Hough transform procedure [70] was added to find the rotation of the license plate and to derotate it between the extraction and segmentation stages.

A pair of Google Glass was used to capture images of 42 separate vehicles. Each vehicle was captured from 5 different positions; three front images from varying distance between 1.5 meter and 5 meter as well as two from an angle of up to 10 degrees. Each image was resized to a resolution of 1920x1080 to match the maximum preview frame resolution that could be provided by the Google Glass camera. For each of the ALPR stages the configurations from the stage optimization experiments were used; 640x360 pixels input frame for the vertical edge detection, 1.13 as scaling factor for the classifier search-window and maximum available plate and character height for the segmentation and recognition stages (but not lower than 20 pixels and 15 pixels respectively).

The implementation was feed the 210 input frames to test success rate and execution time on the ARM-based smart glasses. The license plate was only considered correctly recognized if all six characters were successfully and correctly output from the system.

Chapter 5

Results

5.1 Stage Optimization

5.1.1 License Plate Extraction

Each input frame from the dataset was tested for each of the 30 individual configurations. For each configuration and frame, the total execution time was recorded as well as if a license plate was successfully extracted. A plate was considered successfully recorded only if a clean derotated license plate with some additional padding was the output.

The results from the extraction stage optimization of the license plate extraction module are visualized in Figure 5.1. The bars illustrate the recognition rate for each of the 30 configurations with regards to execution time, success rate divided by execution time. Full details can be found in Appendix A.1.

The vertical edge detection performs extremely bad for the 320x180 pixel input resolution with a recall just above 4%. The recall rapidly improves for input resolutions higher than 320x180 and peaks at slightly above 98%, but at the cost of execution time. At even higher resolutions the performance decrease slightly and the execution time increase rapidly. The chart clearly shows that the optimal performance is achieved with configuration 12; 640x360 pixels input size and a scale factor of 1.13.

5.1.2 Character Segmentation

Each of the license plate images from the dataset were resized to each of the five defined input heights then feed to the segmentation stage. The execution time and success rate

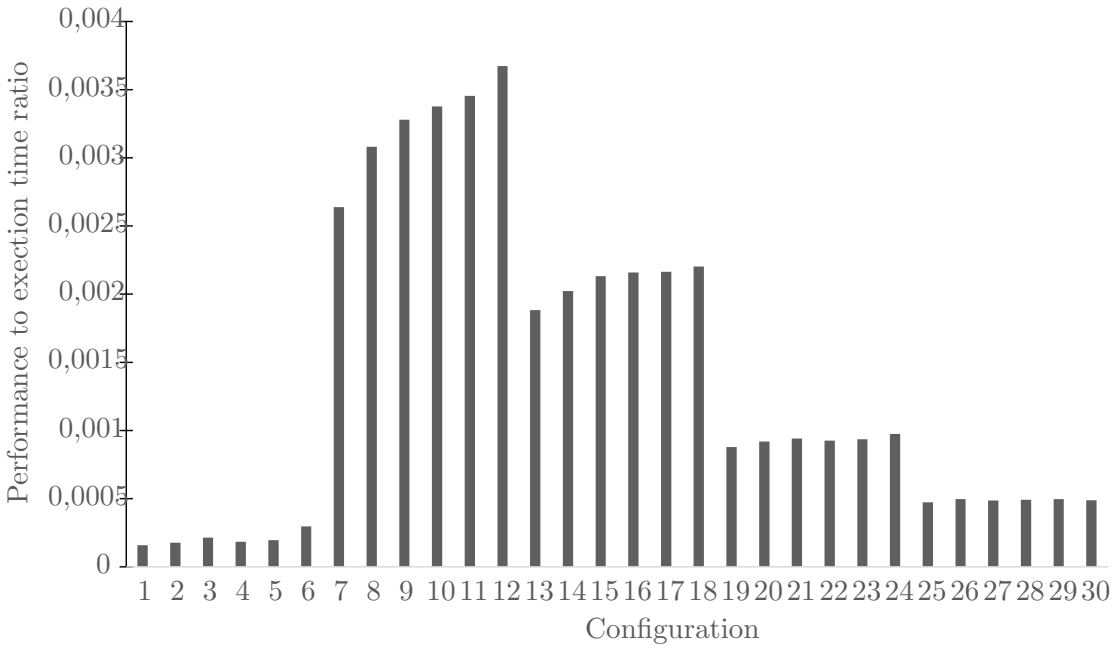

FIGURE 5.1: A chart of the ratio, success rate divided by execution time, for configuration 1 through 30. Taller bars represent higher success to time ratio.

were recorded. The license plate was only considered successfully segmented if all 6 of the plate's characters were correctly segmented.

Figure 5.2 presents the results from the character segmentation optimization. Each bar corresponds to the height of the input license plate and the percentage of plates in which all six characters were correctly segmented and output.

The result is remarkably high and might to some extent be due to the somewhat artificial nature of the manually cropped license plate input images as compared to the slightly sloppier cropping performed by the extraction stage. Figure 5.2 clearly indicates that there is a minimum plate height of 20 px required for successful segmentation. The execution time for each of the input heights were below 1 ms.

5.1.3 Character Recognition

Each of the character segments in the dataset were resized to each of the four defined character heights and feed to the recognition stage. Execution time and success rate were recorded for each scenario. A character was considered successfully recognized if the correct digitized character was output from the recognition stage.

The results in Figure 5.3 reflects the performance of the character recognition stage for the various four input character heights. Each bar corresponds to one of the four character heights together with its respective recognition rate.

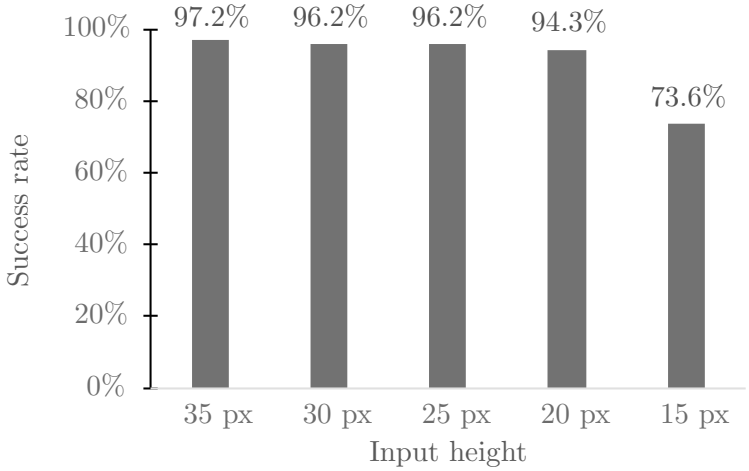

FIGURE 5.2: A chart of the percentage successfully segmented license plates as a result of input license plate height.

As with the segmentation stage there seem to be a clear minimum of 15 pixels hight needed for the recognition stage to successfully recognize each character. For larger inputs the recognition rate is very good. It performs marginally worse than found in [40], which is expected due to the fact that the training was done with just one typeface. The character height feed to the recognition stage had very low impact on the execution time as all recognitions took around 4 ms to perform.

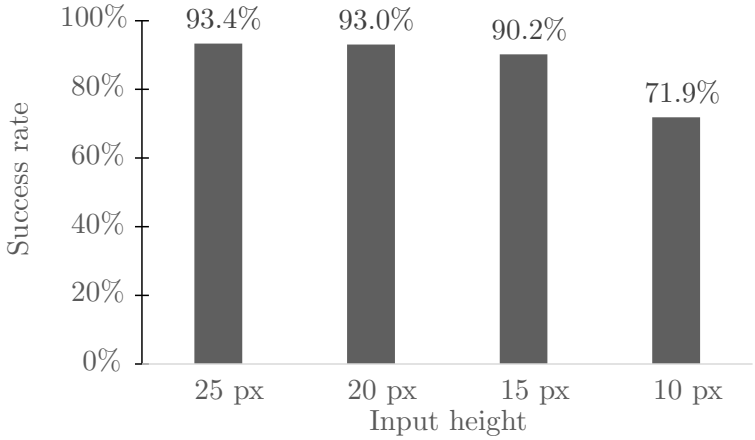

FIGURE 5.3: A chart of the percentage successfully recognized characters as a result of input character height.

5.2 Feasibility study

For the feasibility study each of the stages were linked together, and for robustness a derotation procedure was added between the extraction and segmentation stages. The configuration from previous experiments was used. The system was fed with real-world images from the dataset and the success rate as well as execution time were recorded. A

license plate was only considered successfully recognized if all 6 characters were correctly output in digitized text.

Table 5.2 indicates the percentage of all license plates that made it successfully through each individual stage and Table 5.1 reflects the success rate of each individual stage, e.g the percentage of license plates fed into a stage that were successfully output from that stage.

The system success rate is very good, 79.5%, when considering the limitation in the training of the recognition stage. It is easily derived that license plates containing the letters "W" and "M" reduce the success rate greatly. If plates containing the characters "W" and "M" are removed, the system performs with 87.4% success rate.

	All plates	**Plates without W and M**	**Plates containing W and M**
Extraction success	96.7%	97.4%	90.0%
Segmentation success	95.6%	95.1%	100.0%
Recognition success	86.0%	94.3%	5.6%

TABLE 5.1: The individual success rate for each stage for three scenarios

	All plates	**Plates without W and M**	**Plates containing W and M**
Correctly extracted	96.7%	97.4%	90.0%
Correctly segmented	92.4%	92.6%	90.0%
Correctly recognized	79.5%	87.4%	5.0%
Average execution time	320 ms	316 ms	359 ms

TABLE 5.2: The percentage of input images that were successfully output from each stage and average total execution time for three scenarios.

Chapter 6

Discussion

6.1 Stage Optimization

6.1.1 License Plate Extraction

As mentioned in Section 5.1.1, the performance when using 320x180 pixel as frame size for the vertical edge detection is exceptionally low. As other research [18] have achieved good or at least decent results while applying a similar resolution, one would expect much better performance. To understand why this is, the wide-angle nature of the camera fitted to current smart-glasses need to be taken in to consideration. When analyzing the type of images used in previous research it quickly becomes obvious that the input frames used in those studies are taken with a much more narrow-angle lens and/or at a closer distance. Both of which lead to the license plate appearing larger in the image, and thus keep their vertical edges even though the frame resolution is low. As can be derived from Table A.1, in this study we need at least an input resolution of 640x360 pixels in order to successfully detect license plate candidates at up to 5 meters distance and with the Google Glass camera as capturing device.

FIGURE 6.1: A comparison of an image from a previous study and an image from this thesis. From left to right: An input image captured at close distance and with a narrow-angle capturing device. An image captured using Google Glass.

The slight loss in recall for input sizes larger than 800x600 pixels is likely because of the increase in details and noise, thus producing a large set of potential false plate candidates. This is confirmed by checking the number of license plate candidates generated for each input resolution, as the number of candidates rapidly increase for larger input frame resolutions.

There is no significant difference in recall, around 4 percentage units between the highest and lowest scaling factor, when varying the scaling factor, but a significant percentage impact on the total execution time for the lower input resolutions. As the size of the license plate candidates are not related to the input size used for the vertical edge detection, it is completely natural that the time consumption is independent of the input resolution, thus affecting the total execution time of low input resolutions more than the larger resolutions.

6.1.2 Character Segmentation

By studying the results in Figure 5.2 it is obvious that the character input height need to be higher than 15 pixels in order to achieve good segmentation results. It is further clear that the execution time is very fast, averaging 1 millisecond, for each of the tested input sizes, thus the extracted license plate should be kept with as high resolution as possible in order to produce good segmentation results.

When studying the actual output from the module it is easy to see that when the input resolution decrease, the thresholding procedure starts eliminating important character features due to its small size. The characters also tend to approach a thickness of 1 px or less, which further makes the module fail, due to the fact that one black pixel is allowed in the white lines of the license plate. With such thin character, sometimes the thresholding makes the characters break apart, resulting in character being split up into multiple segments, hence making the segmentation fail.

This method of segmentation is very sensitive to rotation and noise in the image and therefore produces bad results for e.g. small or very dirty license plates. As mentioned in Section 4.2.4, it was chosen for it speed, and not necessarily for robustness. In hindsight a CCA procedure might have been more robust, but at the time it was discarded in favor for the faster projection approach.

6.1.3 Character Recognition

The performance of the character recognition module is very good, as can be seen in Figure 5.3, while taking into consideration that it is a general purpose library. There is

an apparent drop in performance for characters with a height less than 15 pixels, but for larger characters the module produces recognition rates of well above 90%. There are no significant difference in execution time between the tested input resolutions, which indicate that the highest available resolution should be passed to the character recognition module.

When analyzing what characters the module fails at recognizing there are two letters (W and M) and three digits (9, 6 and 3) that are more common than the others.

Studying the W and M characters, it is clear that these two characters are most dissimilar in both the new and the old license plates when compared to the Alte DIN 1451 Mittelschrift, which was used for training the OCR engine. Further, all instances of the digits 9, 6 and 3 that were misrecognized belong to license plates with the new Swedreg typeface. In the Swedreg typeface both the 9 and the 6 have lost some of the curvature in their "tail" and "top" as well as the 3's "top" have become completely flat, as can be seen in Figure 6.2. This makes them distinctly different from the training samples and thus hard to for the OCR stage to recognize. Being able to train the character recognition stage using both the Alte DIN 1451 Mittelschrift typeface and the Swedreg typeface would most likely improve the result significantly.

FIGURE 6.2: The characters 3, 6 and 9 in the new Swedreg typeface (left) and the characters 3, 6 and 9 in the Alte DIN 1451 Mittelschrift typeface (right).

6.2 Feasibility study

The overall system performance, P_{system}, can be approximated from the previous module tests as equation 6.1.

$$P_{system} = P_{extraction} \times P_{segmentation} \times P_{recognition}^{6} \tag{6.1}$$

Based on the performance from each of the previous tests and the equation 6.1, the over all system performance should be 58.7% when applying the best performing configurations. In Table 5.2 the actual over all system performance is indicated as 79.5% when tested on on the 210 images spread across 42 vehicles, which is well above the estimation. This means that the system is performing 20.8 percentage points above the expected performance.

To see why this is, the individual module performance is listed in Table 5.1 and compared to the expected performance in Table 6.1. In the comparison it is evident that the majority of the performance gain is archived in the recognition module. There are many reasons why this could be, but the most likely is the usage of characters with greater height. In the module test all of the input characters where of the same height, between 10 pixels and 25 pixels. The system implementation always utilize the maximum resolution available, thus as the test images are taken at different distances and angles from the vehicles resulting in the character resolution varying between 19 pixels and 46 pixels. This result emphasizes the importance of a high resolution input for the recognition module to perform optimally.

	Expected performance	**Measured performance**
Extraction module	91.1%	96.7%
Segmentation module	97.2%	95.6%
Recognition module	66.3% (93.4% per char)	86.1%

TABLE 6.1: A comparison between calculated system performance and measured system performance

As was mentioned in Section 6.1.3, the letters M and W in the Swedish license plate typeface are significantly different from the Alte DIN 1451 Mittelschrift used to train the recognition stage. Table 5.2 reveal that only 5% of the license plates containing either of the characters M or W were successfully recognized. Removing all images with plates containing these two letters improved the over all system performance by 7.9 percentage points, resulting in a total performance of 87.4% for the complete system. This also suggests that the system performance could be significantly improved by obtaining and training the OCR engine with the actual license plate typeface.

The execution time archived is no where close to being able to process all frames produced by the camera (30 FPS). In Table 5.2 the average execution time to process one frame is calculated to 320 ms, which corresponds to just above 3 FPS, but from a user interaction perspective the necessary response time is much lower. Ickin et al. [71] recommend a response time of less 950 ms for mobile applications to ensure user satisfaction and the 320 ms is well below this threshold. This means that almost three frames can be processed before the user need a response from the application. These frames could e.g. be combined into one output and in turn increase the confidence in the recognition.

Further, during the heavy load of the benchmark the Google Glass became noticeably warmer and the battery performance reduced greatly. Even though there are available external battery packs to tackle the battery issue, this might indicate that the Google Glass specifically is not yet ready for intensive and demanding industry use, but these limitations might not apply to other smart glasses brands.

When forcing the Google Glass preview frame size (the camera stream used to grab individual frames for the live stream) to its maximum resolution 1920x1080, to be able to detect the license plate at as far distance as possible, it struggled to update the screen in a smooth manner. This further adds to the immaturity concerns of the Google Glass.

Many of the hardware limitations will most likely be reduced as dedicated graphical processing units become available to developers. As many of the computations performed in the proposed system are suitable for GPU's, these will be able to significantly reduce the load on the CPU and most likely speed up the execution while reducing battery consumption at the same time.

Chapter 7

Conclusion

During the study of previous research, a combination of vertical edge detection for license plate candidate detection and a boosted cascade classifier using Haar-like features was deemed the fastest and most accurate method for license plate extraction on ARM-based smart glasses. Further, the extremely fast character segmentation method of vertical and horizontal projections was chosen, as well as the widely used and open source OCR library, Tesseract, for character recognition.

The ALPR system was successfully implemented on a pair of ARM-based Google Glass smart glasses with 79,5% successfully recognized license plates while processing slightly above 3 FPS on average. Even though the system performs well enough for non-critical applications there is a large room for improvement and as discussed in Section 6.1.3 there are indications that a large performance gain could be achieved by simply recreating the actual license plate typefaces to train the recognition stage with.

7.1 Future research

When the GPU becomes more developed and accessible to developers on smart glasses, there could most likely be a significant boost in execution time, as many of the computations in the suggested implementation can be performed independently. This should be researched as well as how such an implementation should be implemented.

Another interesting aspect would be to research the possibility to pass heavy computations through a wireless connection to the users smartphone, which most likely have more computational power available. If latency is of low importance, then pushing the computations to the cloud would be an interesting aspect to research.

While performing this thesis, it was obvious that there is very little knowledge about how information should be presented in this new format. This needs to be research as well as aspects on how the users would like to interact with the device. During some shorter sessions where unexperienced smart glass users got to try the device, it was apparent that the way to interact with the glasses was not intuitive.

It also needs to be researched and determined wether smart glasses are safe to wear while operating a vehicle.

Bibliography

[1] Gartner. Gartner predicts by 2017, 30 percent of smart wearables will be inconspicuous to the eye, 2014. URL `http://www.gartner.com/newsroom/id/2941317`.

[2] Jiri Matas and Karel Zimmermann. Unconstrained licence plate and text localization and recognition. In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 225–230. IEEE, 2005.

[3] Yo-Ping Huang, Chien-Hung Chen, Yueh-Tsun Chang, and Frode Eika Sandnes. An intelligent strategy for checking the annual inspection status of motorcycles based on license plate recognition. volume 36, pages 9260–9267. Elsevier, 2009.

[4] Chung-Mu Hwang, Shyh-Yeong Shu, Wen-Yu Chen, Yie-Wern Chen, and Kuang-Pu Wen. Pc-based car license plate reader. In *Applications in Optical Science and Engineering*, pages 272–283. International Society for Optics and Photonics, 1992.

[5] Peter Davies, Neil Emmott, and Nick Ayland. License plate recognition technology for toll violation enforcement. In *Image Analysis for Transport Applications, IEE Colloquium on*, pages 7–1. IET, 1990.

[6] David J Robertson. Automatic number plate reading for transport applications. In *Electronics in Managing the Demand for Road Capacity, IEE Colloquium on*, pages 13–1. IET, 1993.

[7] Muhammad Sarfraz, Mohammed Jameel Ahmed, and Syed A Ghazi. Saudi arabian license plate recognition system. In *Geometric Modeling and Graphics, 2003. Proceedings. 2003 International Conference on*, pages 36–41. IEEE, 2003.

[8] Halina Kwaśnicka and Bartosz Wawrzyniak. License plate localization and recognition in camera pictures. In *3rd Symposium on Methods of Artificial Intelligence*, pages 243–246. Citeseer, 2002.

[9] Shan Du, Mahmoud Ibrahim, Mohamed Shehata, and Wael Badawy. Automatic license plate recognition (alpr): A state-of-the-art review. volume 23, pages 311–325. IEEE, 2013.

[10] Shen-Zheng Wang and Hsi-Jian Lee. A cascade framework for a real-time statistical plate recognition system. volume 2, pages 267–282. IEEE, 2007.

[11] David Llorens, Andrés Marzal, Vicente Palazón, and Juan M Vilar. Car license plates extraction and recognition based on connected components analysis and hmm decoding. In *Pattern Recognition and Image Analysis*, pages 571–578. Springer, 2005.

[12] Abdul Mutholib, Teddy S Gunawan, Jalel Chebil, and Mira Kartiwi. Development of portable automatic number plate recognition system on android mobile phone. In *IOP Conference Series: Materials Science and Engineering*, volume 53, page 012066. IOP Publishing, 2013.

[13] Wisam Al Faqheri and Syamsiah Mashohor. A real-time malaysian automatic license plate recognition (m-alpr) using hybrid fuzzy. volume 9, pages 333–340, 2009.

[14] Kap Kee Kim, KI Kim, JB Kim, and Hang Joon Kim. Learning-based approach for license plate recognition. In *Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop*, volume 2, pages 614–623. IEEE, 2000.

[15] Bai Hongliang and Liu Changping. A hybrid license plate extraction method based on edge statistics and morphology. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 831–834. IEEE, 2004.

[16] Kazumasa Miyamoto, Kazuo Nagano, Mitsuaki Tamagawa, Ichiro Fujita, and Masayuki Yamamoto. Vehicle license-plate recognition by image analysis. In *Industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON'91., 1991 International Conference on*, pages 1734–1738. IEEE, 1991.

[17] Christoph Busch, R Domer, Christian Freytag, and Heike Ziegler. Feature based recognition of traffic video streams for online route tracing. In *Vehicular Technology Conference, 1998. VTC 98. 48th IEEE*, volume 3, pages 1790–1794. IEEE, 1998.

[18] Danian Zheng, Yannan Zhao, and Jiaxin Wang. An efficient method of license plate location. volume 26, pages 2431–2438. Elsevier, 2005.

[19] Luis Salgado, Jose M Menendez, Enrique Rendon, and Narciso Garcia. Automatic car plate detection and recognition through intelligent vision engineering. In *Security Technology, 1999. Proceedings. IEEE 33rd Annual 1999 International Carnahan Conference on*, pages 71–76. IEEE, 1999.

[20] R Parisi, ED Di Claudio, G Lucarelli, and G Orlandi. Car plate recognition by neural networks and image processing. In *Circuits and Systems, 1998. ISCAS'98. Proceedings of the 1998 IEEE International Symposium on*, volume 3, pages 195–198. IEEE, 1998.

[21] Ming G He, Alan L Harvey, and Paul Danelutti. Car number plate detection with edge image improvement. In *Signal Processing and Its Applications, 1996. ISSPA 96., Fourth International Symposium on*, volume 2, pages 597–600. IEEE, 1996.

[22] Hamid Mahini, Shohreh Kasaei, and Faezeh Dorri. An efficient features-based license plate localization method. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pages 841–844. IEEE, 2006.

[23] Tran Duc Duan, Duong Anh Duc, and Tran Le Hong Du. Combining hough transform and contour algorithm for detecting vehicles' license-plates. In *Intelligent Multimedia, Video and Speech Processing, 2004. Proceedings of 2004 International Symposium on*, pages 747–750. IEEE, 2004.

[24] Tran Duc Duan, TL Hong Du, Tran Vinh Phuoc, and Nguyen Viet Hoang. Building an automatic vehicle license plate recognition system. In *Proc. Int. Conf. Comput. Sci. RIVF*, pages 59–63, 2005.

[25] Wangchao Le and Shaofa Li. A hybrid license plate extraction method for complex scenes. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pages 324–327. IEEE, 2006.

[26] Kuan Zheng, Yuanxing Zhao, Jing Gu, and Qingmao Hu. License plate detection using haar-like features and histogram of oriented gradients. In *Industrial Electronics (ISIE), 2012 IEEE International Symposium on*, pages 1502–1505. IEEE, 2012.

[27] Xiaowei Xu, Zhiyan Wang, Yanqing Zhang, and Yinghong Liang. A method of multi-view vehicle license plates location based on rectangle features. In *Signal Processing, 2006 8th International Conference on*, volume 3. IEEE, 2006.

[28] Mei-Sen Pan, Jun-Biao Yan, and Zheng-Hong Xiao. Vehicle license plate character segmentation. volume 5, pages 425–432. Springer, 2008.

[29] Mei-Sen Pan, Qi Xiong, and Jun-Biao Yan. A new method for correcting vehicle license plate tilt. volume 6, pages 210–216. Springer, 2009.

[30] Yungang Zhang and Changshui Zhang. A new algorithm for character segmentation of license plate. In *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*, pages 106–109. IEEE, 2003.

[31] Ying Wen, Yue Lu, Jingqi Yan, Zhenyu Zhou, Karen M von Deneen, and Pengfei Shi. An algorithm for license plate recognition applied to intelligent transportation system. volume 12, pages 830–845. IEEE, 2011.

[32] Charl Coetzee, Charl Botha, and David Weber. Pc based number plate recognition system. In *Industrial Electronics, 1998. Proceedings. ISIE'98. IEEE International Symposium on*, volume 2, pages 605–610. IEEE, 1998.

[33] Nobuyuki Otsu. A threshold selection method from gray-level histograms. volume 11, pages 23–27, 1975.

[34] Takashi Naito, Toshihiko Tsukada, Keiichi Yamada, Kazuhiro Kozuka, and Shin Yamamoto. Robust license-plate recognition method for passing vehicles under outside environment. volume 49, pages 2309–2319. IEEE, 2000.

[35] Shigueo Nomura, Keiji Yamanaka, Osamu Katai, Hiroshi Kawakami, and Takayuki Shiose. A novel adaptive morphological approach for degraded character image segmentation. volume 38, pages 1961–1975. Elsevier, 2005.

[36] Xifan Shi, Weizhong Zhao, and Yonghang Shen. Automatic license plate recognition system based on color image processing. In *Computational Science and Its Applications–ICCSA 2005*, pages 1159–1168. Springer, 2005.

[37] Zhang Sanyuan, Zhang Mingli, and Ye Xiuzi. Car plate character extraction under complicated environment. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 5, pages 4722–4726. IEEE, 2004.

[38] Cheokman Wu, Lei Chan On, Chan Hon Weng, Tong Sio Kuan, and Kengchung Ng. A macao license plate recognition system. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 7, pages 4506–4510. IEEE, 2005.

[39] Cemil Oz and Fikret Ercal. A practical license plate recognition system for real-time environments. In *Computational Intelligence and Bioinspired Systems*, pages 881–888. Springer, 2005.

[40] Lihong Zheng and Xiangjian He. Character segmentation for license plate recognition by k-means algorithm. In *Image Analysis and Processing–ICIAP 2011*, pages 444–453. Springer, 2011.

[41] JAG Nijhuis, MH Ter Brugge, KA Helmholt, JPW Pluim, L Spaanenburg, RS Venema, and MA Westenberg. Car license plate recognition with neural networks and fuzzy logic. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 5, pages 2232–2236. IEEE, 1995.

[42] Abdulkerim Capar and Muhittin Gokmen. Concurrent segmentation and recognition with shape-driven fast marching methods. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 155–158. IEEE, 2006.

[43] Kwang-Baek Kim, Si-Woong Jang, and Cheol-Ki Kim. Recognition of car license plate by using dynamical thresholding method and enhanced neural networks. In *Computer Analysis of Images and Patterns*, pages 309–319. Springer, 2003.

[44] Vojtěch Franc and Václav Hlaváč. License plate character segmentation using hidden markov chains. In *Pattern Recognition*, pages 385–392. Springer, 2005.

[45] Eun Ryung Lee, Pyeoung Kee Kim, and Hang Joon Kim. Automatic recognition of a car license plate using color image processing. In *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, volume 2, pages 301–305. IEEE, 1994.

[46] Lu Xiaobo, Ling Xiaojing, and Huang Wei. Vehicle license plate character recognition. In *Neural Networks and Signal Processing, 2003. Proceedings of the 2003 International Conference on*, volume 2, pages 1066–1069. IEEE, 2003.

[47] F Aghdasi and H Ndungo. Automatic license plate recognition system. In *Proc. AFRICON Conf. Africa*, volume 1, pages 45–50, 2004.

[48] Mi-Ae Ko and Young-Mo Kim. A simple ocr method from strong perspective view. In *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, pages 235–240. IEEE, 2004.

[49] Peifeng Hu, Yannan Zhao, Zehong Yang, and Jiaqin Wang. Recognition of gray character using gabor filters. In *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, volume 1, pages 419–424. IEEE, 2002.

[50] Siti Norul Huda Sheikh Abdullah, Marzuki Khalid, Rubiyah Yusof, and Khairuddin Omar. License plate recognition using multi-cluster and multilayer neural networks. In *Information and Communication Technologies, 2006. ICTTA'06. 2nd*, volume 1, pages 1818–1823. IEEE, 2006.

[51] Yafeng Hu, Feng Zhu, and Xianda Zhang. A novel approach for license plate recognition using subspace projection and probabilistic neural network. In *Advances in Neural Networks–ISNN 2005*, pages 216–221. Springer, 2005.

[52] Fikriye Öztürk and Figen Özen. A new license plate recognition system based on probabilistic neural networks. volume 1, pages 124–128. Elsevier, 2012.

[53] Alte din 1451 mittelschrift font family, 2015. URL `http://www.1001fonts.com/alte-din-1451-mittelschrift-font.html`.

[54] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[55] Christopher H Messom and Andre L Barczak. Stream processing for fast and efficient rotated haar-like features using rotated integral images. volume 7, pages 40–57. Inderscience, 2009.

[56] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–900. IEEE, 2002.

[57] Rashed Mustafa, Yang Min, and Dingju Zhu. Obscenity detection using haar-like features and gentle adaboost classifier. volume 2014. Hindawi Publishing Corporation, 2014.

[58] Opencv library, 2015. URL `http://opencv.org/`.

[59] Scott Krig. Imaging and computer vision resources. In *Computer Vision Metrics*, pages 411–418. Springer, 2014.

[60] Tesseract ocr library, 2015. URL `https://code.google.com/p/tesseract-ocr/`.

[61] Chirag Patel, Atul Patel, and Dharmendra Patel. Optical character recognition by open source ocr tool tesseract: A case study. volume 55, pages 50–56. Foundation of Computer Science, 244 5 th Avenue,# 1526, New York, NY 10001, USA India, 2012.

[62] Tesseract ocr library wiki, 2015. URL `https://code.google.com/p/tesseract-ocr/wiki/TrainingTesseract3`.

[63] Mei Yu and Yong Deak Kim. An approach to korean license plate recognition based on vertical edge matching. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 4, pages 2975–2980. IEEE, 2000.

[64] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. volume 30, pages 32–46. Elsevier, 1985.

[65] Flickr, 2015. URL `http://www.flickr.com`.

[66] Uc irvine machine learning repository, 2015. URL `http://archive.ics.uci.edu/ml/`.

[67] Rainer Lienhart, Alexander Kuranov, and Vadim Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *Pattern Recognition*, pages 297–304. Springer, 2003.

[68] Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. volume 14, pages 771–780. JAPANESE SOC ARTIFICIAL INTELL, 1999.

[69] Sandip Rakshit and Subhadip Basu. Recognition of handwritten roman script using tesseract open source ocr engine. 2010.

[70] Jiri Matas, Charles Galambos, and Josef Kittler. Robust detection of lines using the progressive probabilistic hough transform. volume 78, pages 119–137. Elsevier, 2000.

[71] Selim Ickin, Katarzyna Wac, Markus Fiedler, Lucjan Janowski, Jin-Hyuk Hong, and Anind K Dey. Factors influencing quality of experience of commonly used mobile applications. volume 50, pages 48–56. IEEE, 2012.

Appendix A

Module optimization results

A.1 License Plate Extraction Results

Config.	Input resolution	Scaling factor	Plate candidates	Recall	Total execution time
1	320x180	1.03	1,2	4.03%	255 ms
2	320x180	1.05	1,2	3.23%	183 ms
3	320x180	1.07	1,2	3.23%	151 ms
4	320x180	1.09	1,2	2.42%	131 ms
5	320x180	1.11	1,2	2.42%	124 ms
6	320x180	1.13	1,2	3.23%	109 ms
7	640x360	1.03	6,1	94.4%	358 ms
8	640x360	1.05	6,1	94.4%	306 ms
9	640x360	1.07	6,1	93.6%	285 ms
10	640x360	1.09	6,1	91.9%	272 ms
11	640x360	1.11	6,1	91.1%	264 ms
12	640x360	1.13	6,1	91.1%	248 ms
13	800x600	1.03	9,4	98.4%	523 ms
14	800x600	1.05	9,4	98.4%	486 ms
15	800x600	1.07	9,4	98.4%	462 ms
16	800x600	1.09	9,4	96.8%	448 ms
17	800x600	1.11	9,4	96.0%	444 ms
18	800x600	1.13	9,4	94.4%	429 ms
19	960x720	1.03	18,8	95.2%	1084 ms
20	960x720	1.05	18,8	94.4%	1026 ms
21	960x720	1.07	18,8	93.6%	993 ms

22	960x720	1.09	18,8	91.9%	994 ms
23	960x720	1.11	18,8	91.9%	981 ms
24	960x720	1.13	18,8	92.7%	951 ms
25	1280x960	1.03	25,4	93.6%	1973 ms
26	1280x960	1.05	25,4	93.6%	1882 ms
27	1280x960	1.07	25,4	91.1%	1870 ms
28	1280x960	1.09	25,4	91.9%	1869 ms
29	1280x960	1.11	25,4	91.9%	1852 ms
30	1280x960	1.13	25,4	88.7%	1817 ms

TABLE A.1: The complete results from the license plate extraction optimization.