LPR: License Plate Recognition Project

Final Report

Altynbek Amirzhanov | Yeskhat Kuat | Abylay Omar

The purpose of this report is to explain the implementation of our project, “License Plate Recognition”. This report will begin with sections on motivation, past projects, and constraints. It will then proceed to describe our system in broad terms to provide a general overview of our project to the reader. The report ends with a discussion on possible future work using more evolved image processing techniques and acknowledgments.

# Introduction

Monitoring vehicles for law enforcement and security purposes is a difficult problem because of the number of automobiles on the road today. An example is this lies in border patrol: it is time consuming for an officer to physically check the license plate of every car. Additionally, it is not feasible to employ a number of police officers to act as full-time license plate inspectors. Police patrols cannot just drive in their cars staring at the plates of other cars. There must exist a way for detecting and identifying license plates without constant human intervention. As a solution, we wanted to implement a system that can extract the license plate number of a vehicle from an image – given a set of constraints.

# Past Projects

Although there have been similar projects in the past, we have been able to find a few research papers on the topic of license plate recognition systems as well as several commercial license plate recognition systems. Among the research papers: medium.com, researchgate.com, and etc.

# The Constraints

Due to the limited amount of time we have, a set of constraints have been placed on our system to make the project more manageable, they are as follows:

* Image of the vehicle taken from a fixed angle.
* Image of the vehicle taken from a fixed distance.
* The vehicle is stationary when the image was taken.
* Standard Kazakhstan license plates will be given by users.

# Project Overview



Our license plate recognition system can be roughly broken down into the above block diagram. The image acquisition is done by images that were taken from the Internet and stored on the PC; an interface on the PC side then transfers the image from the PC to the VM (virtual machine), where the actual image processing occurs; at the end of image processing, the system returns to the PC the license plate number of the vehicle in the image. The details of each subsystem are to follow in the following sections.

# The main part

We have used OpenCV and HaarCascade models in our project. We have trained our model with Russian license plates from open-source data. Before training, we did pre-processing for our dataset. The main principle of HaarCascade is that CascadeClassifier takes one object as a positive example and others as a negative example. So, when we train our model with pictures that contain license plates it counts as positive and others as negative. To check our model we have used data from kolesa.kz as a testing phase. It gave us errors in some cases but in most of the cases, our model worked correctly.

# Image acquisition

The images of vehicles were taken from the Internet, with different resolutions. On average, the images were taken seven feet away from the vehicle. They were stored in color JPEG or PNG format on the websites. We use Python to convert the color JPEG, PNG images into grayscale raw format on the PC. An interface on the PC side then transfers the images to the VM for processing.

# Image Processing Overview

Once the grayscale image is received by the VM from the PC, the image processing process begins. Image processing can be further broken down into three phases: candidate selection, candidate verification, and optical character recognition. First, the system uses edge detection to compile a list of candidates. Second, it chooses a candidate to be the license plate based on discrimination operations performed on the original grayscale image. Lastly, optical character recognition was performed on this region designated as the license plate. The details of the subsystems are discussed in the following sections.

# Display Results

Our demonstration showcased how the output for the license plate recognition system can look. It basically had a prominent display of what the system considers to be the license plate characters, along with several likely candidates in the event that the license plate was incorrectly identified. The candidates are chosen depending on the specific characters on the plate since there are some characters that are more easily confused than others. Additional output for the system is the license plate candidate that the system identified. This was given in the form of a binary file with binary pixel values.

The demonstration was in the form of a simple command-line style output. This can easily be ported into a more graphical interface simultaneously displaying the license plate image, the input image, and the license plate. The dashes represent “wild-card” operators where previously established characters can be placed based on position.

# Future Work

Due to time constraints and the lack of experience in image processing in our group, we are unable to make this license plate recognition system as functional as it could be. There are numerous improvements that could be made, such as:

* Use more evolved image processing techniques to improve the accuracy of the system. For example, we could use binary morphology to eliminate edges that are thinner than the characters of the license plate.
* Expand the system to work with variable angles and distances accurately.
* Expand the system to work with non-Russian License Plates, as we trained our model with Russian license plates.

# Acknowledgments

Code:

* We got most of the codes on the websites like kolesa.kz, medium.com, <https://habr.com/post/208092/>, [https://github.com/MicrocontrollersAndMore/OpenCV\_3\_License\_Plate\_Recognition\_Python](https://github.com/Microcontrollers%20AndMore/OpenCV_3_License_Plate_Recognition_Python), etc.

Ideas:

* This first idea was provided by Altynbek Amirzhanov.
* Yeskhat Kuat and Abylay Omar have improved the output idea.

Assistance:

* Special thanks to Yeskhat and Abylay for developing most of the project.