Ukrainian Catholic University

Faculty of Applied Sciences

License Plate Recognition Implementation and Optimization

Linear Algebra final report

Authors:

Volodymyr Savchuk

Oleksandr Kukhar

Danylo Kalchenko

Supervisor:

Nadiya Lyaskovska

12 May 2021

1. Introduction

In the modern world traffic congestion is one of the central problems for the majority of big as well as small cities. Every day the quantity of cars on the roads is growing, as a result, the numbers of violations are increasing, the numbers of stealing cars cases are growing. Shortage of parking places is the reason for unauthorized cars to enter the private zones in finding free parking spaces.

Automated number plate recognition system(ANPR) is the key solution to the problems listed before. Such a system will be able to detect license plates automatically, and send fines to those who violate traffic rules, the number of accidents on roads will be much less. Also, such a system could be useful in the vehicles' autopilots, however, the system must be as accurate as possible, because people's lives depend on that.

Research area of our project is object recognition and computer vision. In order to create such a system we need to construct an algorithm, which will, firstly, identify the location of the number plate in the frame, extract the characters from it, and recognize them. The aim of the project is to create a program that would be able to detect and recognize license plates from photos and videos, it can be easily performed in the primitive case when a photo is clear, and license plate is well visible, but it is really hard to recognize that from any photo, and our goal is to make it as accurate as it is possible.

2. Problem Setting

Suppose we are given a snapshot from the videocam on a motorway. The idea of our program is to proceed with this picture and return the license plate number as a result. License Plate algorithm consists of three stages: Number plate detection, Character segmentation, Character recognition. Each of the stages has different approaches of implementation(different algorithms).

We decided to concentrate on and analyze the first stages, namely, Number plate detection. So the problem becomes smaller: our program should proceed with the

snapshot from the videocam on a motorway and extract only the number plate region, namely, the four points coordinates of the rectangle, where the number plate contains.

3. Overview of approaches

Detection of the number plate is a crucial task, because Character segmentation and Character recognition fully rely on its accuracy. We have analyzed different methods for the detection of number plate such as:

- ***Edge Detection based Plate Extraction*** proposed in [1] is a double chance algorithm, because it uses Line grouping(LD) and Edge Density (ED). Firstly, LD is used and it determines line segments and groups it by geometrical conditions. Then the algorithm leaves only rectangular forms, because they are in our field of interest. If there are no such forms then the algorithm proceeds with ED. ED defines the plate region where the vertical lines have the biggest density. Using a real-life database collected by the speed enforcement camera, a high success rate of 99.45% was obtained[1].

- **Edge Detection based on Canny Algorithm** is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It consists of 5 stages: Noise reduction using Gaussian filter, Gradient calculation along the horizontal and vertical axis, Non-Maximum suppression of false edges, Double thresholding for segregating strong and weak edges, Edge tracking by hysteresis.

- ***Texture based Candidate Extraction***

The method uses projection to detect the candidate number plate. Since the maximum projection does not correspond to the plate area all the time, several peak lines in vertical projection are detected and maximum peak values are considered as reference lines. After binarization, all connected component areas across each reference line are sorted by acreage, which is used to calculate the size of the sliding window. All candidates are extracted along with Rank filter and Robert's operator.

Also, we decided to deeply analyze algorithms defining which points create lines.

- **Lines Detection with Hough Transform** estimates the two parameters that define a straight line. The transform space has two dimensions, and every point in the transform space is used as an accumulator to detect or identify a line described by

$$r = x * cos(\theta) + y * sin(\theta)$$

In practice, as input data the algorithm uses the edged(binary) image for simplifying calculations. Therefore we should take into account the imperfection of the edged picture like distorted pixels, which cause non solid contours. That is the reason for choosing Hough transformation as it provides a flexible interface. While implementing it we can individually define some constraints on shapes and so that improve the quality of detection. Among the cons of Hough transformation is its slight straightforwardness and therefore plenty of calculations.

- **Line Detection with Convolution-based technique:** the line detector operator consists of convolution masks tuned to detect the presence of lines of a particular width n and a θ orientation. Here are the four convolution masks to detect

horizontal, vertical, oblique (+45 degrees), and oblique (−45 degrees) lines in an image.

4. Algorithm analysis

- **Canny Edge Detection**

Canny Edge Detection algorithm is composed of 5 steps:

1. Noise reduction
2. Gradient calculation
3. Non-maximum suppression
4. Double threshold
5. Edge Tracking by Hysteresis

1. Noise reduction

The reason for noise reduction is that edge detection technique is very sensitive to image noise. One of the ways to get rid of noise on the image is to use Gaussian blur in order to smooth it. Gaussian blur is done using the convolution technique with a Gaussian Kernel. The equation for a Gaussian filter kernel of size $(2k+1)\times(2k+1)$:

$$H_{ij} = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}\right)$$

2. Gradient calculation

The gradient calculation part detects the edge intensity and direction. In general, edges correspond to a change of pixels' intensity. To detect such a change a the easy way is to apply the Sobel filter to the image. To apply the Sobel filter we need to convolve the matrix of x's and y's derivatives with the Sobel K_x

kernels and K_y respectively.

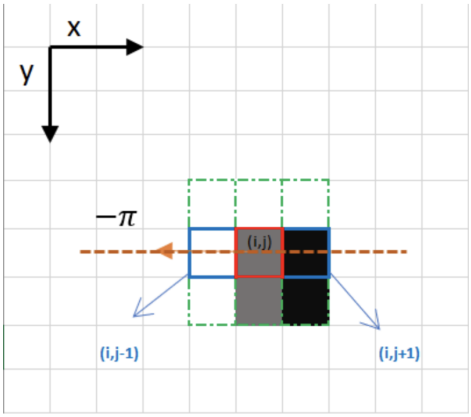
Then $K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$ the magnitude $K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$ G and the slope Θ are calculated in a such way:

$$|G| = \sqrt{I_y^2 + I_x^2}$$

$$\theta(x, y) = \arctan\left(\frac{I_y}{I_x}\right)$$

3. Non-Maximum Suppression

The principle is that the algorithm goes through all the points in the gradient intensity matrix and finds the points with maximum value in edge directions.



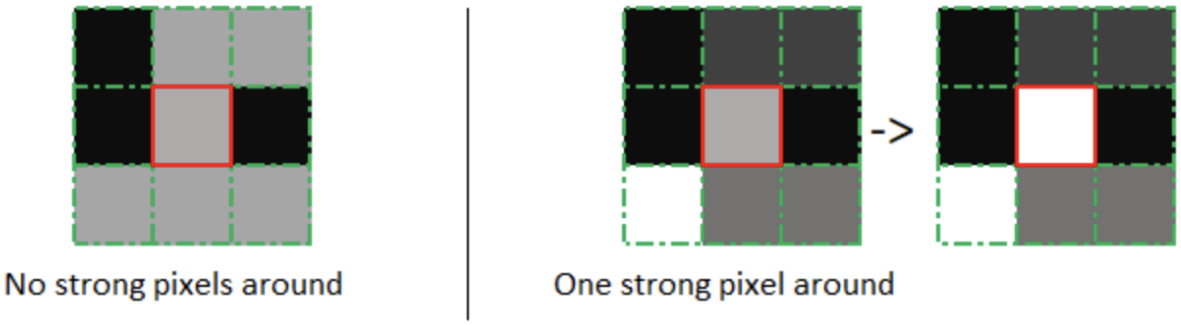
For example, in the picture above the edge direction is the orange dotted line. The algorithm should check if the pixels in the same direction are more intense or not than the present pixel. If there is a more intense pixel, then this pixel is kept and others become black. If there are no more intense pixels, then the present pixel is kept and others become black. In the picture above pixel (i, j-1) is kept and pixel (i, j) is set to 0.

4. Double Threshold

At this step we will identify 3 types of pixels: strong, weak and non-relevant. Strong have such strong intensity that they definitely contribute to the final edge. Weak pixels have not enough big intensity to be considered definitely contributing to the final edge and at the same time not enough small intensity to be considered non-relevant. All other pixels are non-relevant.

5. Edge Tracking by Hysteresis

At this step we will use the threshold results to make from weak pixels strong ones. Here we just go through all pixels in the image and look where at least one of the pixels around the pixel being processed is a strong, then the pixel being processed changes for a strong one.



5. Conclusions

In this project we successfully implemented Canny Edge Detection algorithm and Hough Transform algorithm. In such a way we realized the first part of the Automated number plate recognition system, namely, Number plate detection.

The code is available on GitHub

Literature:

[1] - Shen-Zheng Wang, His-Jian Lee, “Detection and recognition of license plate characters with different appearances,” Proc. of 16th International Conference on Pattern Recognition, 2003, vol.3, pp. 979-983.

[2] - Xiangjian He et al, “Segmentation of characters on car license plates,” 10th Workshop on Multimedia Signal Processing, Oct. 2008, pp. 399-402.

[3] - Feng Yang, Zheng Ma, Mei Xie, “A Novel approach for license plate character segmentation,” ICIEA, 2006, pp.1-6.

[4] - Canny Edge Detection - Retrieved from: https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123

[5] - Lines Detection with Hough Transform - Retrieved from: https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549