FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Detection and Reading Number Plates in Photos
Student:	Patrik Vodila
Supervisor:	Ing. Miroslav Čepek, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2022/2023

Instructions

Review literature dealing with identification and reading of number plates in pictures. After consultation with the supervisor choose datasets of pictures with number plates (for example, from Kaggle) and test selected techniques to assess their performance on selected datasets. Explore ways to improve the performance of the selected techniques. Also, assess if the selected techniques are applicable in different use-cases (eg. form reading).

Electronically approved by Ing. Karel Klouda, Ph.D. on 24 February 2022 in Prague.

Bachelor's thesis

DETECTION AND READING NUMBER PLATES IN PHOTOS

Patrik Vodila

Faculty of Information Technology
Katedra teoretické informatiky
Supervisor: Ing. Miroslav Čepek, Ph.D.
May 12, 2022

Contents

List of Figures

List of Tables

I would like to thank my supervisor, Ing. Miroslav Čepek, Ph.D., for the opportunity to write such exciting work. I appreciate your willingness to help to achieve this ultimate goal.
I want to thank the Faculty of Information Technology CTU in Prague and all employees of this faculty for their valuable guidance throw all these hardworking years.
Furthermore, I would like to thank my family and my loved ones for all the support needed throw the study.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Praze on May 12, 2022 .....

Abstrakt

Práce se zabývá automatickým rozpoznáváním registračních značek vozidel. Výsledkem této práce je program, který je schopen rozpoznávat text na jednořádkových registračních značkách s tmavým textem na světlém pozadí. Výstupem programu je fotografie s označenými všemi detekovanými registračními značkami i s rozpoznaným textem. Program je napsán v programovacím jazyce Python a používá YOLO verzi 4 implementovanou za pomoci TensorFlow pro lokalizaci značky a Tesseract pro rozpoznávání znaků. Použitím knihovny OpenCV moje implementace zlepšila počátečné výsledky používající pouze Tesseract OCR a YOLO. Zlepšení správného rozpoznávání bylo více než desetkrát lepší. Správné rozpoznání značek dosáhlo až do 31,5%. Našel jsem také více případů použití, kde se implementace osvědčila.

Klíčová slova umělá inteligence, počítačové vidění, strojové učení, rozpoznávání objektů, optické rozpoznávání znaků, automatizované čtení registračních značek vozidel

Abstract

The work deals with the automatic recognition of vehicle number plates. The result of this work is a program that is able to recognize text on a single-line number plate with dark text on a light background. The output of the program is an image marked with all detected number plates and with recognized text. The program is written in Python and uses YOLO version 4 implemented using TensorFlow for number plate localization and Tesseract for character recognition. By using the OpenCV library, my implementation improved the results initially, using only Tesseract OCR and YOLO. The improvement in the correct recognitions was more than ten times better. Correct number plate recognition reached up to 31.5%. I have also found more applicable use-cases where the implementation proved successful.

Keywords artificial intelligence, computer vision, machine learning, object recognition, optical character recognition, automatic license plate recognition

List of abbreviations

AI	Artificial Intelligence
ALPR	Automatic License Plate Recognition
ANN	Artificial Neural Network
ANPR	Automatic Number Plate Recognition
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
EU	European Union
GPU	Graphics Processing Unit
ML	Machine Learning
YOLO	You Only Look Once

Chapter 1
Introduction

We live in a time when the majority of citizens of well-developed countries want to own a vehicle, and many of them own more than one. This fact can cause a number of problems that different regulations must address. For example, every commonly used motor vehicle must have a number plate. Number plates are used to identify and distinguish vehicles.

In the olden days, only people could identify a vehicle. As we well know, people make mistakes. Nowadays, we no longer need to rely on human intelligence to recognize text on the number plate. We have created machines to do so.

In the 21st Century, we have seen a major improvement in information technology. Higher quality cameras, more powerful computers, and more optimal processing helped us use these programs in multiple use-cases, such as number plate recognition.

Automatic number plate recognition is already widely used. Many police officers now have access to a spectrum of valuable devices that helps them identify vehicles with a camera. There are also many parking lots where the computer allows us into the parking lot according to the vehicle number plate within seconds.

Although it is clear that the issue of number plate recognition is not new, and this problem is already solved by many commercial and non-commercial solutions, this topic is of particular interest to me because it deals with computer vision. And with the growing number of images, image processing is a rising field of computer vision.

My goal is to create an application that will recognize the text on the number plate. My program will recognize the most widely used number plate type: one line of dark text (usually black or dark blue) with light background (usually white or yellow).

Before I reach the key objective of creating the application, I must first review the literature dealing with automatic number plate recognition in images. Then I will choose an image dataset with a number plate. I will also analyze available techniques useful for my solution.

After a successful implementation, I will test the application on the selected dataset. And finally, I will try to use my application for other use-cases.

Chapter 2
Analysis

This chapter will analyze the problem of automatic number plate recognition. Next, I will define the necessary terms in this section, with whom I will be working throughout the whole thesis. I will also try to show some examples of number plates across the world. I will also try to go through the theory briefly.

2.1 Number plates

This chapter will look at the ANPR (automatic number plate recognition) problem and the various limitations and difficulties I have faced while creating my implementation. Next, I will focus on the diversity of number plate names in different parts of the world. Later, I will go through the legislation and other international agreements which govern the Czech Republic. Then I will go through specific symbols in the number plate that can cause problems when trying to solve ANPR. Finally, I will focus on the types of number plates that will be recognized with my implementation.

2.1.1 Number plate naming convention across the world

Number plate is "a sign on a road vehicle that shows its registration number"[1]. Usually, the plate has information such as registration number, region identification, information about the last emission control, and the last technical inspection of the vehicle.

Although several countries use a similar type of vehicle identification, the naming convention between these countries is different. In the United Kingdom, the phrase number plate is used. For the United States of America, the license plate is being used. Furthermore, in the case of European countries, registration plate is often used.

For this reason, various terms and abbreviations are used in different parts of the world. For example, the acronym ANPR (automatic number plate recognition) is utilized in the United Kingdom. And a similar abbreviation with the equivalent meaning ALPR (automatic license plate recognition) is used in the United States of America.

Nevertheless, the meaning remains the same.

2.1.2 Various number plates across the world

There are a large number of different layouts and types of number plates. An example of different sizes used worldwide are in Figure 2.1.

Figure 2.1 Example of different sizes[2]

Figure 2.2 License plate in Nepal[3]

A certain number plate is occasionally visibly divided into sections separated by a line, for example, in Figures 2.3, 2.4, and 2.5. This can help recognize a specific type of number plate. However, in my implementation, I am unable to take advantage of this line separation.

Also, for example, in Figures 2.2, 2.3, 2.4, and 2.5, different character sets are being used. Adding support for more characters would hurt the performance because every new character allowed on the number plate increases the possibility of error.

There are also number plates with multi-line designs, often combined with different individual characters' sizes, for example, in Figures 2.3 and 2.5.

Unfortunately, I will not be able to recognize such a large number of tags in my implementation. Therefore in Section 2.1.5, I will define a portion of number plates that I will try to recognize with the greatest possible accuracy.

I have even more examples of Czech registration plates in Section 2.1.3.

Figure 2.3 Registration plate in Iran[4]

Figure 2.4 Registration plate in Iraq[5]

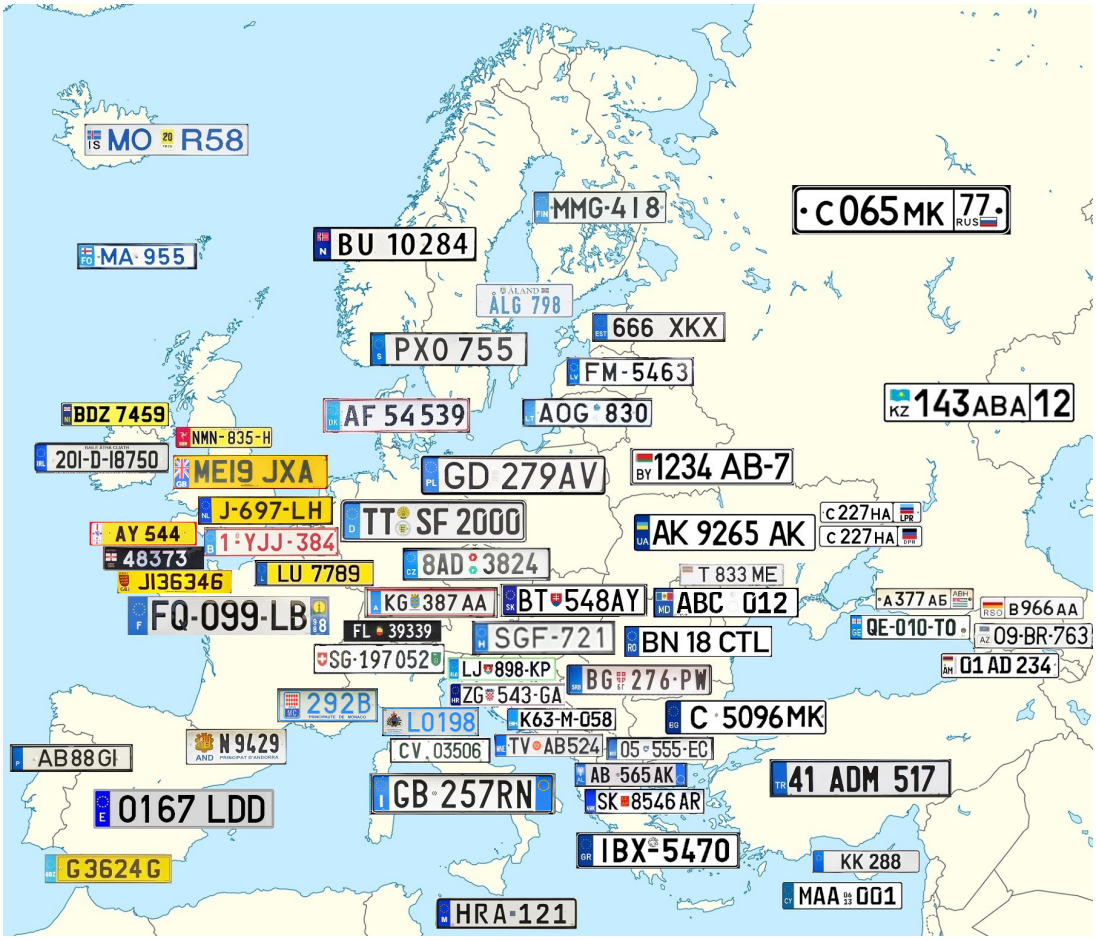

Figure 2.5 Registration plates in Europe from 2020[6]

2.1.3 Czech registration plate

I will take a closer look at the registration plates in the Czech Republic. I will take data from the currently active Decree [7]. The Czech Republic, as a European Union Member State, must also meet the requirements of the EU. For example, Legislation[8].

The Czech Republic has a legally defined range of alphanumeric characters that can be used on a registration plate. These are numbers from "0" to "9" and some capital letters from the Czech alphabet. Letters "G", "CH", "O", "Q", and W are not listed. This set of characters is defined only by the government of the Czech Republic and therefore does not apply worldwide. For example, the Slovak Republic has the character O allowed but has a defined way to separate the letter "O" from the number "0". In Figure 2.6, the allowed alphanumeric characters and their font is to be seen.

Figure 2.6 Alphanumeric characters with the proper font used in the Czech registration plates[7, p. 59]

The Czech Republic also has different layouts. One categorization is by size. The Czech Republic nowadays uses the following six sizes of the registration plate in use:

- 520 x 110 mm
- 340 x 200 mm
- 280 x 200 mm
- 320 x 160mm
- 200 x 160 mm
- 80 x 100 mm

These sizes can be seen in Figure 2.7.

The second division is by color and position. There are seven different layouts in use that can be seen in Figures 2.7, 2.8, 2.9, 2.10, 2.11, 2.12, and 2.13.

2.1.4 Additional disruptive elements

Unfortunately, the difficulty of ANPR problems does not end there. In addition to poor image quality, noise, and sludge, there are additional disruptive elements. Other factors that can make ANPR a challenging task are stickers, national or regional symbols, and other elements of this nature. All these elements are sometimes located on the number plate. The problem is with the fact that the program is unable to filter out everything. Suppose the implementation is not able to filter out all contours that are not part of the symbols used for the vehicle identification. The result of the program for the number plate can have extra symbols, and therefore identification will not be correct.

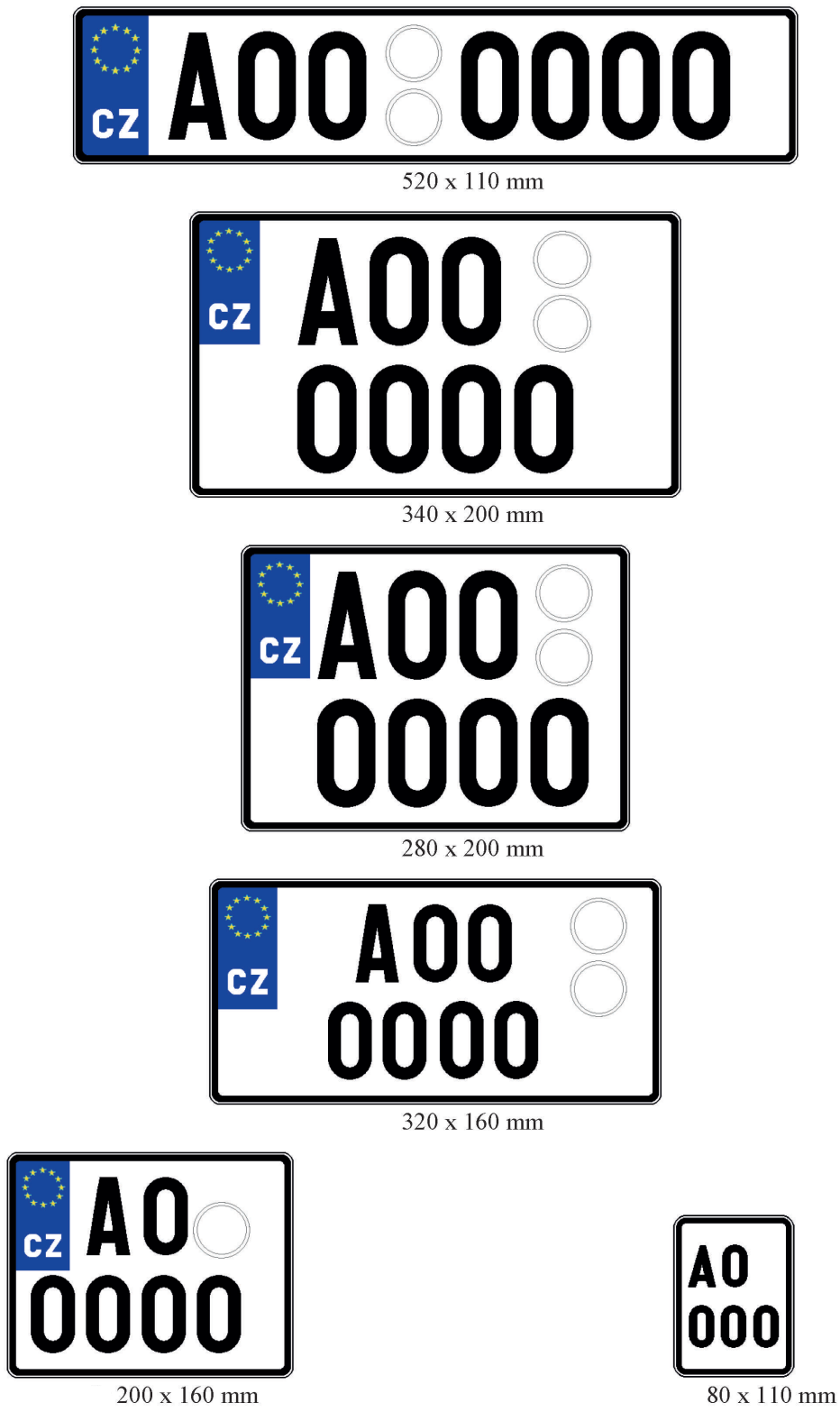

Figure 2.7 Czech registration plate layout 1 with different sizes[7, p. 59]

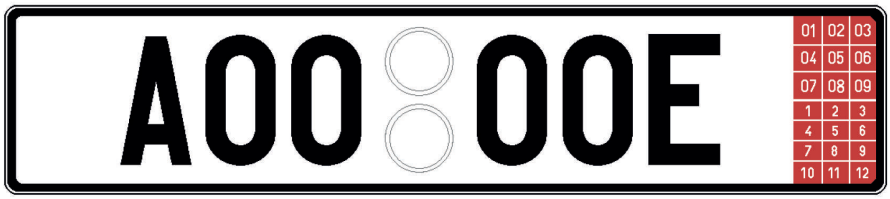

Figure 2.8 Czech registration plate layout 2[7]

Figure 2.9 Czech registration plate layout 3[7]

Figure 2.10 Czech registration plate layout 4[7]

Figure 2.11 Czech registration plate layout 5[7]

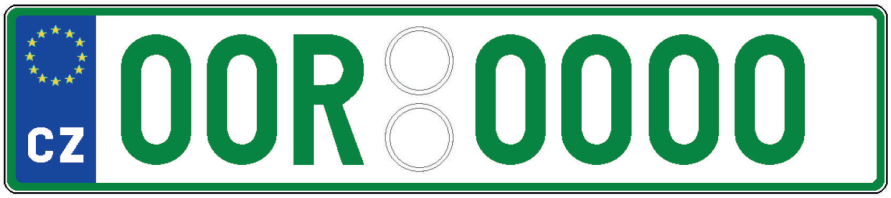

Figure 2.12 Czech registration plate layout 6[7]

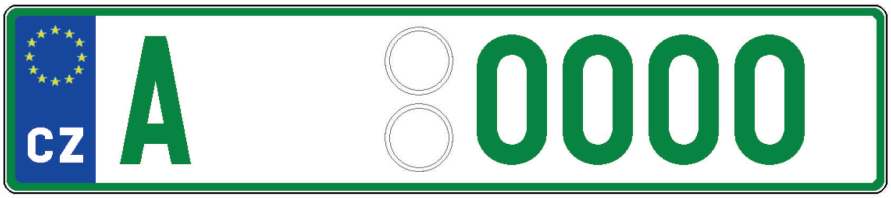

Figure 2.13 Czech registration plate layout 7[7]

Another fact that is very harmful to ANPR is using the number plate external attachment to secure the number plate in place. Many times driver tries to secure the number plate by zip ties. The legality of these actions is questionable because recognition using specialized cameras that use the reflexivity of number plates is quite impaired.

2.1.5 Problem conclusion

Every state in the world has the right to create its own rules and define its own way of identifying vehicles. Even countries belonging to the European Union have their own rules, such as the positioning of characters on the number plate, preset of characters being used on the number plate, and font used for the symbols.

Among other things, most states have several types of number plates and therefore creating ANPR for all cases with an almost perfect detection rate becomes an impossible task.

Ideally, ANPR is specified for a special type of number plate in a particular region.

My solution will focus on recognizing number plates with a one-line design where there is dark text on a light background.

2.2 AI, ML, ANN, and DL

In today's world, the terms: Artificial intelligence, Machine learning, Artificial neural network, and Deep learning are often referred to as modern things, often used as the selling point of a product. Although these terms are considered modern, they were formed much sooner than many think. The first mention dates back to the summer of 1956 by John McCarthy at a conference.[9]

The opinion of most experts and researchers is that ML is part AI and DL is part ML. I have a diagram presenting this relation in Figure 2.14.

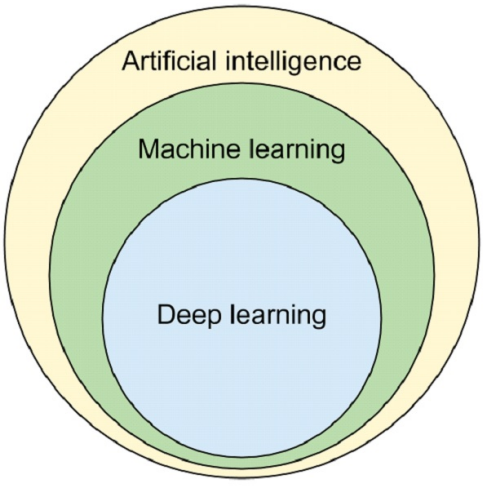

Figure 2.14 Artificial intelligence, Machine learning, and Deep learning diagram[10]

Nevertheless, since then, the minority opinion of researchers from ML has also emerged. They consider ML as a separate field with links to AI. Many of them argue that certain parts of ML can be performed without any AI knowledge.

The most common problem with these terms is their mutual confusion. Therefore, let me go through the definitions.

2.2.1 Artificial intelligence

John McCarthy wrote in his book "WHAT IS ARTIFICIAL INTELLIGENCE?": "It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable."[11, p. 2]

However, a simpler and shorter definition for artificial intelligence is often used. It is a field of computer science, where computers try to do tasks that are attributed to work by intelligent beings.

2.2.2 Machine learning

Exactly the way people have the ability to learn, there is an effort to teach computers how to learn. Therefore, a branch of AI focused on using algorithms to allow computers to get knowledge from available data was created. This branch is called machine learning.[12]

2.2.3 Artificial neural network

The human brain works with the use of neurons. An idea to imitate this to work for computers is called an Artificial neural network. The main idea is to have many neurons communicate with each other, resulting in an educated decision.[13]

In the case of ANN, it has a collection of connected units called Artificial neurons. Usually, those neurons are distributed into multiple layers. There are three types of layers: input, output, and hidden.

The layer that receives input data is called the input layer. The layer that produces the final result is called the output layer. The rest of the layers between input and output layers are called hidden layers. An ANN can have multiple hidden layers.

An example of ANN with two hidden layers is in Figure 2.15.

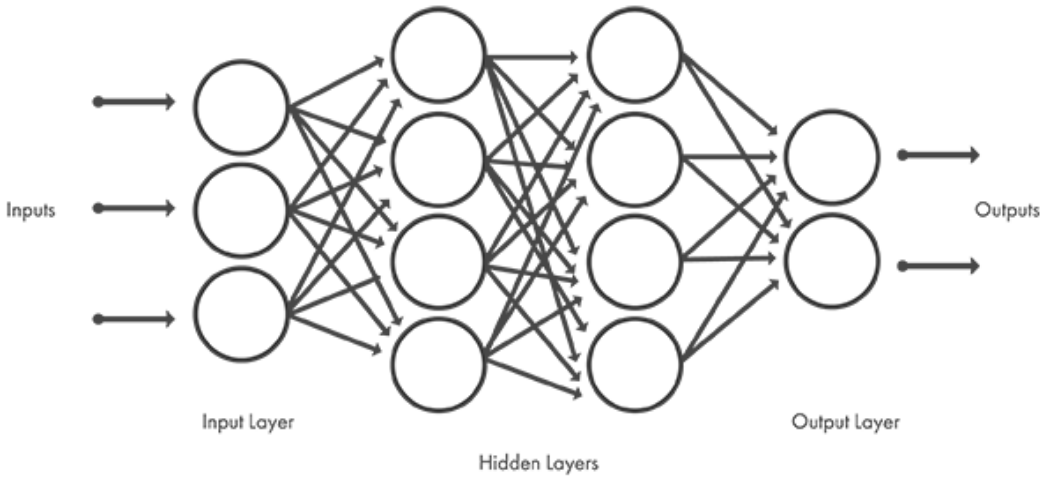

Figure 2.15 Artificial neural network diagram with two hidden layers[14]

2.2.3.1 Convolutional neural network

The ANNs are commonly divided into categories according to which mathematical operations are being used. Convolution is one of the most commonly used operations in Computer vision.

The ANNs with at least one convolution operator are called Convolutional neural networks. The layers of neurons that use convolution are called Convolutional layers.

2.2.4 Deep learning

An Artificial neural network with multiple hidden layers is often referred to as a Deep neural network. The advantage of these Deep neural networks is that no data pre-processing is often required, which would otherwise be necessary. Nevertheless, this advantage does not come without a price. The price of this advantage is that for the same result, Deep neural network training needs more significant amount of data compared to training ANN with only a single hidden layer.

A branch of machine learning that takes care of improving results for Deep neural networks is called Deep learning.

2.2.5 Computer vision

Computer vision is a field of AI, focuses on information inquiring from any digital visual input, for example, photos and videos. "If AI enables computers to think, computer vision enables them to see, observe and understand."[15]

The main goal of computer vision is to supplement the human effort needed to detect the problem according to the picture, identify objects, and more.

Computers are already able to outperform people in some tasks in terms of speed and accuracy.

According to the Report [16] form 2021, in 2020 computer vision market size was valued at USD 11.32 billion and is expected to have a compound annual growth rate of 7.3% from 2021 to 2028.

2.3 Gaussian distribution in 2D

First, let me go through what Gaussian distribution is. Gaussian distribution, is also known as the normal distribution, is a "continuous probability distribution"[17] function for a random variable. Gaussian distribution is often used in statistics but has wider use even in Computer vision.

The general use Gaussian distribution function is defined:

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{(x-\mu)^2}{2\sigma^2}\right)}$$

The parameter μ is the mean of the distribution, x is the random variable, and σ is the parameter of the normal distribution. The σ^2 is the variance of the distribution. The π refers to the mathematical constant approximately equal to 3.141592.

In my case, the mean is equal to zero. Therefore it can be simplified to:

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{x^2}{2\sigma^2}\right)}$$

In the case of two random variables, I get this formula:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$$

The expression can be rewritten:

$$\frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{x^2}{2\sigma^2}\right)} \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{y^2}{2\sigma^2}\right)}$$

The final result of this expression is as follows:

$$G(x,y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{x^2}{2\sigma^2}\right)} \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{y^2}{2\sigma^2}\right)} = G(x)G(y)$$

Furthermore, instead of using complex calculations for normal distributions with two random variables, two separate normal distributions can be used. In our case, even each of these distributions may have a different parameter σ.

The values for such a Gaussian distribution in 2D can be used for applications in image processing.

The Gaussian distribution is well known for the shape called “bell-shaped” curve. In Figure 2.16 is the visualization of why this nickname was utilized.

[18]

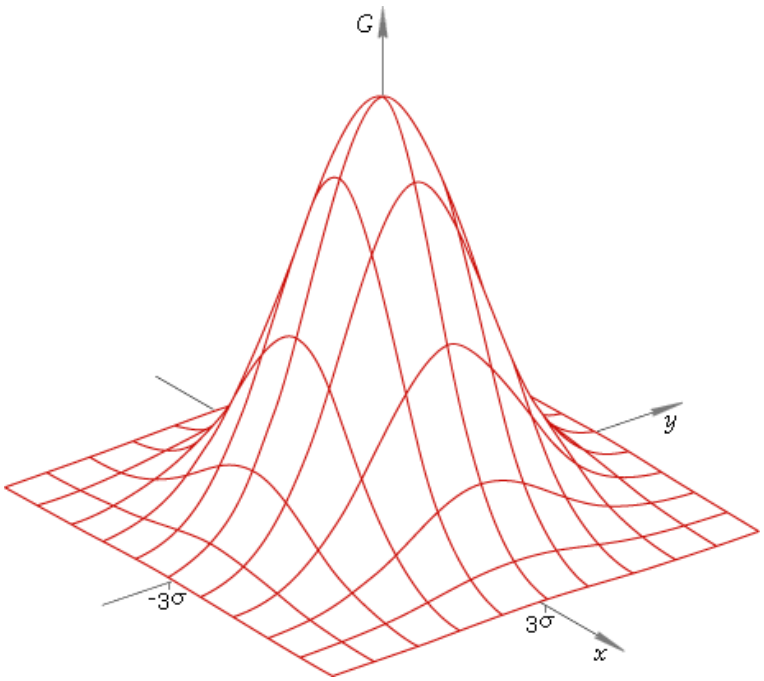

Figure 2.16 2D Gaussian distribution[18]

Chapter 3

Technologies

This chapter will give all the essential information about the technologies and libraries used in this bachelor's thesis.

3.1 Programming language

The ANPR is a problem in a field of AI called Computer vision. For more information about computer vision, go to Section 2.2.5. Mostly recommended programming languages for CV are C++ and Python.

Experts recommend Python as a programming language from the following reasons: [19]

- Easy to use
- Most used programming language for CV
- Excellent documentation
- Good debugging and visualization tools

In my experience, Python is a better programming language for this task. Therefore in my implementation, I will be using Python in version 3.7.

3.2 Object detector

Many object detectors are used. Many of them focus on getting the best results possible. These are some of them: R-CNN, Fast R-CNN, Faster R-CNN, cascade R-CNN, Single Shot MultiBox Detector, Single-Shot Refinement Neural Network, and more.

Although accuracy is an important parameter, someone on the street who wants to recognize a number plate will not have the ability to use the power of a supercomputer. Therefore, I was looking for something for my implementation that would consider both accuracy and speed. The best match was the object detector You Only Look Once.

YOLO is a Deep Neural Network with 26 layers, of which 24 are convolutional. The main goal of this project was to create an ANN that would be able to detect objects in real-time. Over time, with some improvements and newer versions, YOLO is able to compete with the best current models.

3.2.1 Versioning

As with most of the great tools, many try to adapt them for their suite. This is what happened with YOLO, and therefore one simply gets lost among the dozens of YOLO versions.

I have decided to use the primary branch of YOLO. The history of this branch is as follows: YOLO from 2015 [20], YOLO v2 from 2016 [21], YOLO v3 from 2018 [22], and finally, YOLO v4 from 2020 [23], which is still being well maintained at the time of this writing. Every new version was able to come with improvements. Some focus on accuracy and some speed. But the idea is still the same as the first version.

3.2.2 YOLO Model

YOLO divides the input image into an S x S grid. Where S is the size of the input layer. Each grid cell predicts bounding boxes and a confidence score for those bounding boxes. Each bounding cell predicts for every class of object we are detecting the probability of the object being in the bounding box. Figure 3.1 shows the process.[20]

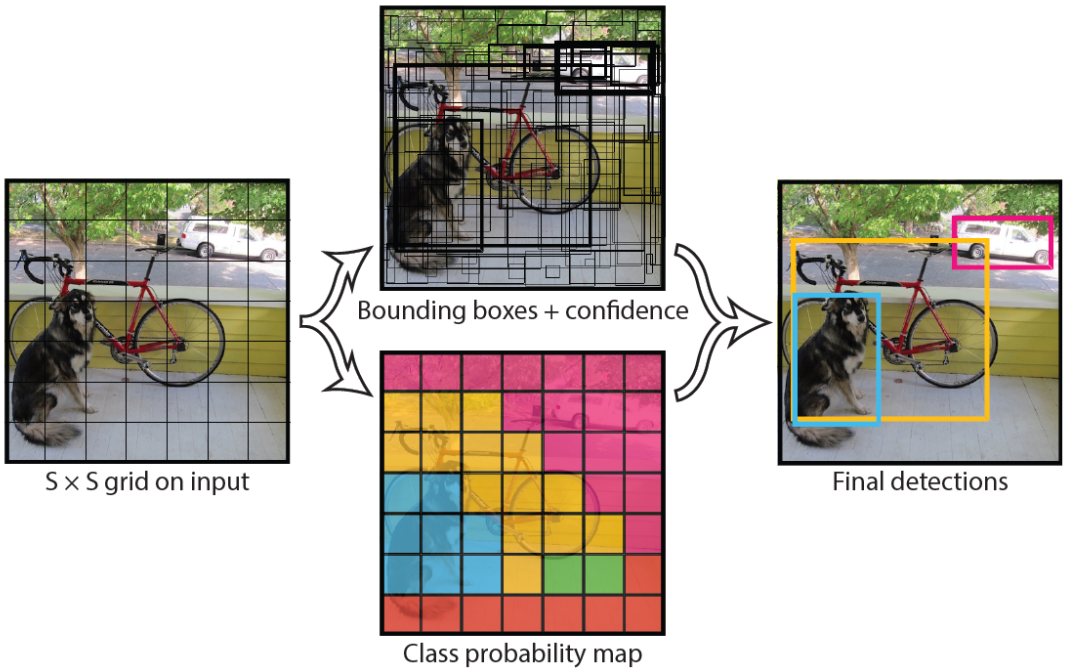

Figure 3.1 The Model of YOLO[20]

In my implementation, I will work with size S equal to 416, and my number of classes is one since I am only detecting number plates. I also decided to use an implementation in TensorFlow. This implementation allows users to use this library not only for desktop computers with dedicated GPU but with some limitations. It can run on devices like smartphones.

3.2.3 Compertion

Figure 3.2 is the comparison of YOLOv4 with other solutions. MS Common Objects in context is used as a validation dataset. A significant increase in precision is shown over a single version.

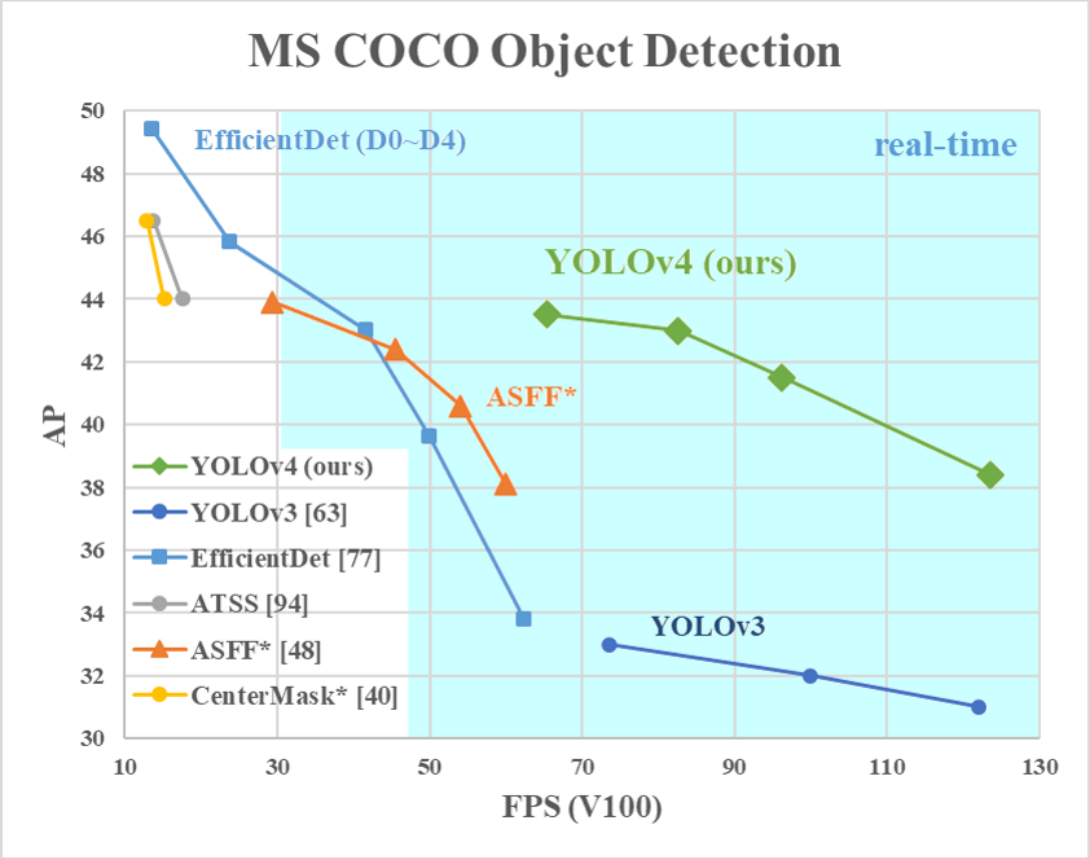

Figure 3.2 Comparison of YOLOv4[23]

3.3 TensorFlow

Among many beginners in AI and especially ML, one of the most popular libraries is TensorFlow. This library makes sure that implementing complex or straightforward ANNs is easy to make. Description from official Github website [24]:

"TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications."

I will be using this platform as the backbone for my computer vision implementation.

3.4 Image processing library

I will be using the OpenCV library for image processing. Where Open stands for open-source and CV stands for Computer Vision. As the name suggests, this is an open-source library for use on Computer Vision. This library is available in several programming languages, including Python.

The library can boast more than 2500 optimized state-of-the-art algorithms for computer vision. As well as algorithms for machine learning.[25]

Some of to features I will be using are image resizing, changing between different color models in the image, blurring, thresholding, and more.

The significant number of users naturally creates a big community with many solutions to every possible problem.

3.5 Tesseract

"Tesseract is ope-source optical character recognition tool. Tesseract has support for Unicode and can recognize words and characters for more than 120 languages." [26]

Tesseract's most vital point is the support of individual languages and their words. Tesseract supports multiple detection modes. Many of them are dedicated to detecting larger blocks of text in specific languages.

Tessecat is a massive and robust tool that tries to do all the work. From line detection and language detection to trying to find individual text characters.[27]

AlthoughTesseract is a powerful tool. Tesseract documentation has a dedicated page on how to improve the result. This page will be my primary source of ideas for improvements. Page is available from[28].

3.6 Python-tesseract

Python-tesseract is a Python library for optical character recognition. Python-tesseract is only a wrapper for Tesseract OCR. The pre-requirement is to have the Tesseract OCR installed and have its path in the PATH environment variable. Depending on the Operating system, installing "tessconfigs" and "configs" may be needed. Refer to the library documentation.[29]

Chapter 4

Image processing

This chapter will describe different algorithms used in our implementation as part of the image processing. I will also describe the mathematics and formulas behind those algorithms.

4.1 Image blurring

Image blurring is an often-used effect in image processing. If I want to draw attention to something, often image blurring is used. This effect causes a specific object to lose a clear and sharp shape. People are used to focusing their view on the sharpest things in the picture, and therefore it can be used to redirect the viewer's viewpoint. Nevertheless, image blurring is also helpful for other use-cases.

I won't be using image blurring to redirect the viewer's viewpoint. In my case, I will use blurring to reduce the noise in the image. Nighttime images can have decreased quality, and therefore the number of disturbing particles that I will want to get rid of will increase. Therefore, our goal of blurring will be to create a picture with a minimal number of disturbing particles while not losing the accuracy of the characters on the number plate.

I had different types of image blurring to choose from. For example: Average blurring, Median blurring, Bilateral filtering, and Gaussian blurring. After several attempts in combination with various thresholding more in Section 4.2. I have achieved the best results with the use of Gaussian blur.

Average blurring is a technique where the center pixel is replaced by averaging all surrounding pixels. In contrast, Gaussian blurring is a weighted average. Values are weighted by the Gaussian distribution in 2D.[30, pp. 86-88] More about the Gaussian distribution is in Section 2.3.

4.1.1 Gaussian blur

Gaussian blur (also known as Gaussian smoothing) is an image blurring technique that simulates the "effect that of viewing the image through a translucent screen"[31].

Gaussian blur is often used as a pre-processing step by Computer Vision Engineers. "If you take a photo in low light, and the resulting image has a lot of noise. Gaussian blur can mute that noise."[32]

In my case, it helps with low light photos as well as with different types of sludge on the number plate. Another benefit of Gaussian blur is that it can help fill possible noise. This noise can cause symbols to be divided into several parts, which could cause a character not to be recognized as one large enough piece for automatic character recognition. Section 4.3 will describe more about searching for characters and their contours.

In the code of my implementation, I am using the library OpenCV for image blurring. More information about the library is in Section 3.4.

This library has implemented the function `GaussianBlur` with main parameters `src` - the input image to be blurred, `ksize` - the size of the square area of the pixel to calculate its value, `sigmaX` and `sigmaY` - the parameter for Gaussian distribution for x and y coordinates.[33]

The result of the blurring techniques for the parameter `ksize = StDev` can be seen in Figure 4.1.

Figure 4.1 Gaussian blur with ksize parameter equal to 3 and 10[34]

4.2 Thresholding

Another image processing method that is widely used in Computer vision is thresholding. The basic idea is to create a binary image using a threshold value. The pixel below the threshold value is set to be zero. And the pixel above the value is set to be a positive number, usually one. Sometimes the inverted result is used instead. This process is called binarization. It is essential to differentiate between grayscale images and binarized images. A grayscale image is an image with colors ranging from black to white. In the case of the binarized image, there are only two values. These values are usually shown in pictures in white and black colors. In many cases, the image is first reduced to the grayscale image before binarization.

There are a few methods for obtaining these binarized images. The simplest method is Simple thresholding, is also known as Global thresholding. This type of thresholding is a simple comparison applied to every pixel. Although this method works well on scenes with low noise and good lighting conditions, it is not sufficient for use in my implementation. Another common method used for thresholding is Otsu's method. This method belongs to the so-called global thresholding methods. However, the global thresholding does not solve the problem of poor lighting and shadows in the photo. Therefore it is not ideal for my use.

The difference between local and global thresholding can be seen in Figure 4.2.

Figure 4.2 Example of global and local thresholding[35]

I achieved the best results using the Adaptive Thresholding methods. More about Adaptive Thresholding methods in Section 4.2.1.

4.2.1 Adaptive Thresholding

This method also has a different activation method. I will only pay attention to the Gaussian method because this gives, according to my experience, the best results.

This method is as good for analyzing sharp edges that are often in the text. Conversely, a problem can occur with large objects. If parameters are not selected correctly, only the object's edge can be visible.

My program code uses the Adaptive Thresholding implementation with a Gaussian activation method in the OpenCV library. More info about the library is in Section 3.4.

This library has implemented the function `adaptiveThreshold` with main parameters: `src` - the input image where to apply the thresholding,`maxValue` - new value for pixel where condition is satisfied,`adaptiveMethod` - the type of activation method, and `blockSize` - size of the square of pixel used to calculate the threshold value.

4.3 Finding contours

Contour is “curve joining all the continuous points”[36]. For finding all the contours in a binary image I am using Topological Analysis by Border Following by Satoshi Suzuki [37].

My program code uses the Finding Conoutor implementation from the OpenCV library. More info about the library is in Section 3.4.

This library has implemented the function `findContours` with the main parameter: `image` - the input binarize image where the contours can be detected.

The function result is a list of all detected contours in the image.

Chapter 5

Implementation

This chapter will define every step of my implementation, the dataset used for evaluating the result as well a reason for the results of my implementation.

My implementation builds on top of a project that implements Yolo in version 4 with the help of TensorFlow. This implementation is open-source under The Massachusetts Institute of Technology License. The YOLO implementation is by hunglc007 and is available at [38]. Also, my implementation uses a pre-trained model. I have the model from the author, theAIGuysCode, and it is available at the address [39]. The current implementation from this author is unfortunately not functional, but I also thank him for some inspiration for my implementation.

5.1 Implementation steps

I was able to divide my implementation into eleven consecutive steps. The individual steps can be seen in Figure 5.1.

The first step of my implementation is detecting all number plates in the image using YOLO version 4. The result is a list of found objects, the probability of belonging to the class of objects, and their position. An example can be seen in Figure 5.2.

From my experience, the YOLO model I used did not have any false positive objects with a probability of over 25%. Therefore, the next step is selecting the largest detected object for the OCR.

In the next step, I will try to prepare a number plate for recognition for Tesseract. The ideal font height for Tesseract is 20 pixels. Since my implementation aims to recognize only one-line tags, the next step is to change the number plate size to the desired size. From my experience, the image with the detected number plate is usually twice as big as the symbols. Therefore new height will be 40 pixels tall, and the width will be changed proportionally. This resizing will help us standardize the data for the following steps.

I know that it is most difficult for a Tesseract to create a binary image so I will help it. However, before I can binarize the image, I have to change it to the grayscale image first. The result of this step is visible in Figure 5.3.

At the moment, I still have noise to remove. Therefore my next step will try to eliminate the noise in the image. The ideal way to eliminate this noise is to use Gaussian Blur. The result of this step is visible in Figure 5.4.

At this point, I got rid of most of the noise, so the image is ready for the next step.

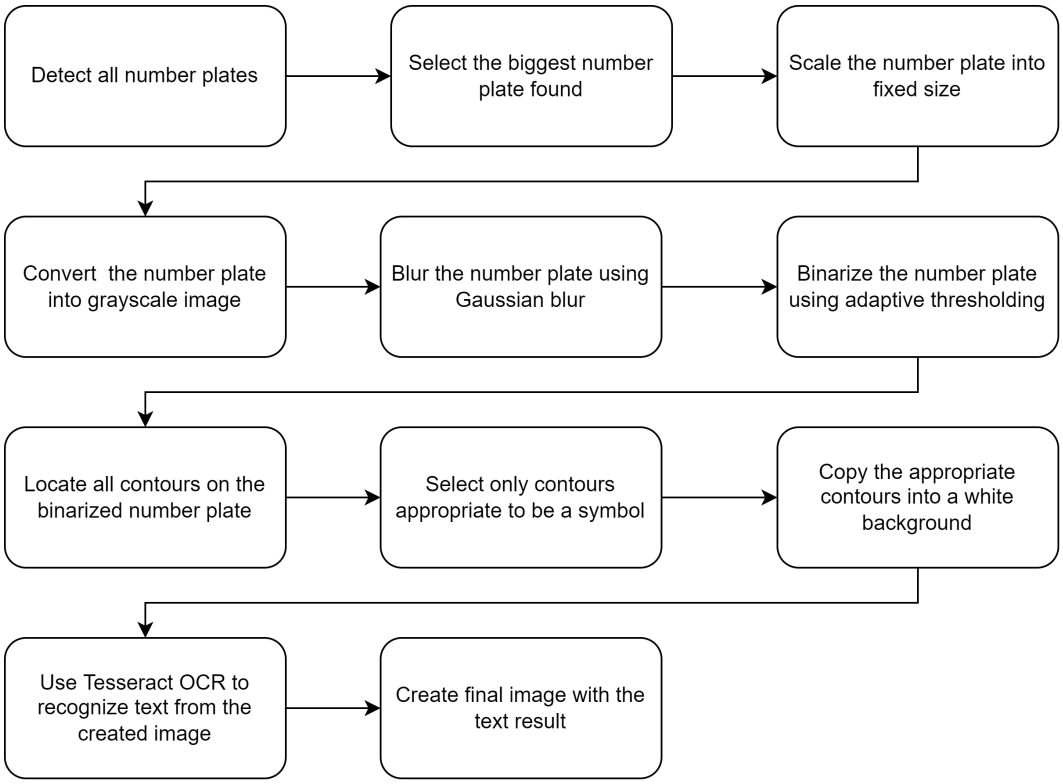

Figure 5.1 Program flow

Figure 5.2 Located number plate

Figure 5.3 Grayscale number plate

Figure 5.4 Blurred number plate

I can finally binarize the image. I will use Adaptive thresholding with the Gaussian activation method for image binarization. As a result of this step, I have a binary image. The result of this step is visible in Figure 5.5.

Nevertheless, my work is not over here. There are still many parts that do not belong to the symbols in these pictures. Therefore, the next step is to detect all contours in the image. I get an extensive contour list where not everything is part of a symbol in this step. The result of this step is visible in Figure 5.6 and 5.7.

In the next step, I will try to filter out all contours that do not have the potential to be a symbol. For this step, I can use the color if the original image is colorful, or I can use the size of the contour. In my implementation, I have both of these examples. I will eliminate parts with the logo of the European Union. I will eliminate objects that do not resemble a letter in size or shape. The result of this step is visible in Figure 5.8 and 5.9.

Next, I will prepare these detected symbols for our OCR by copying the white contours on the black background. The result of this step is visible in Figure 5.10.

I have done everything so that Tesseract OCR can achieve the best possible results.

Therefore, the next step is to call the Tesseract library with a single-line reading mode to detect all characters of the alphabet and all numbers.

The last step is to write the detected character on the original image where I already have the number plate detection shown. The final result is in Figure 5.11.

Figure 5.5 Binarized number plate

Figure 5.6 All detected contours on color number plate

Figure 5.7 All detected contours on binarized number plate

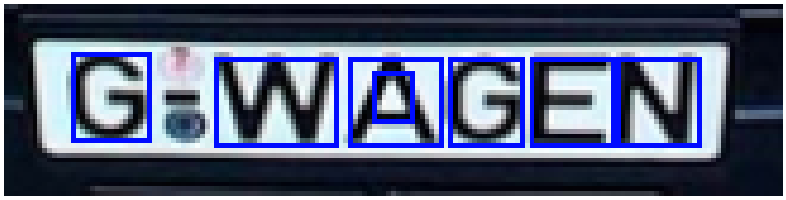

Figure 5.8 Filtered contours on color number plate

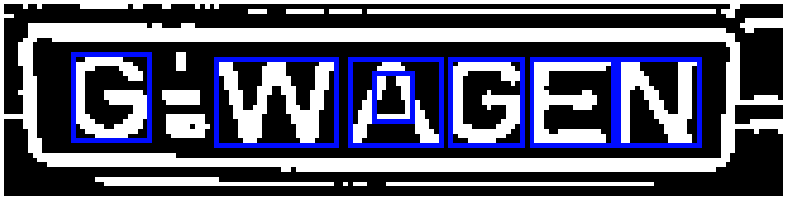

Figure 5.9 Filtered contours on binarized number plate

Figure 5.10 Symbols from number plate

Figure 5.11 Final result

5.2 Dataset

I will use two different datasets to evaluate the results.

The first dataset is freely available and published by OpenALPR to evaluate results from any ANPR system. I will use the European part of this dataset. This dataset is available at [40].

The dataset has color photos with various number plates, both current and some no longer in use. The size and quality vary in the images. There are images in size 2048 x 1536 to the pictures with the size 480 x 360. This dataset also contains a reference result. I will refer to this dataset as EU_dataset.

I managed to get the second dataset from the private company "Villa Pro spoločnosť s ručením obmedzeným" based in Slovakia. This dataset is from a parking lot in Slovakia where a special camera is used. This company has years of experience using ALPR cameras. The cameras are set up precisely so that their software has the best possible results. The shots from the camera are unfortunately not of the best quality. These are 835 x 455 photos embedded in a 1280 x 720 file for support from their software. Another feature of these photos is that they are black and white. The element of reflexivity of the number plates is used. Unfortunately, in my case, the object detector for the number plate is trained on the dataset with colors. Images from this camera are often giving different results than standard cameras, and therefore my object detection tool is not able to recognize the number plate. Also, the dataset contains a large number of Slovak number plates for which Tesseract is unable to identify the character 'O' due to the specific font used. I will refer to this dataset as External_dataset.

5.3 Results

I will compare results from my implementation to results from test implementation with no image processing. The implementation without image processing will only use YOLO to detect the objects and then Tesseract OCR to recognize number plates without any pre-processing. I will refer to the implementation without image processing as my baseline implementation. The resulting percentage will be rounded to one decimal place.

In EU_dataset, the baseline result is, that from 108 photos, Tesseract was only able to recognize 3 number plates, which is 2.8%. My implementation improved the correct recognition by 28.7% to 31.5%, which is more than 11 times better. In baseline implementation, Tesseract was only able to recognize any meaningful text in 53 cases which make 49.1%. I was able to recognize meaningful text in all cases, but one where the number plate was inverted in colors meaning the text was in a light color, and the background was dark. This case was outside of my implementation goal. This means that some meaningful text was extracted in 107 cases out of 108, which make 99.1%. The baseline implementation has the correct number of symbols detected in the number plate only in 8.3% of the cases. My implementation was able to approve it to 64.8%.

Some results for EU_dataset are available in Appendix A.

In External_dataset, the baseline result is, that from 2211 photos, Tesseract was only able to recognize 31 number plates, which is 1.4%. My implementation was able to improve the correct recognition by 14.3% to 15.7%, which is more than ten times better. In baseline implementation, Tesseract was only able to recognize any meaningful text in 558 cases which make 25.2%. I was able to recognize meaningful text in 1938 cases which make 25.2%. In most of these cases, the YOLO could not locate the number plate in the images from the specialized camera. Pre-training the model could significantly increase this number. The baseline implementation has the correct number of symbols detected in the number plate only in 4.9% of the cases. My implementation was able to approve it to 48.4%.

Tables 5.1 and 5.2 show the results for EU_dataset and External_dataset, respectively.

	Correct recognitions	Detected some symbols	Correct number of symbols
Baseline solution	2.8%	49.1%	8.3%
My implementation	31.5%	**99.1%**	**64.8%**

Table 5.1 Results for EU_dataset

	Correct recognitions	Detected some symbols	Correct number of symbols
Baseline solution	1.4%	25.2%	4.9%
My implementation	15.7%	**87.7%**	**48.4%**

Table 5.2 Results for External_dataset

5.4 Different use-cases

Since I decided to implement my implementation with the Tesseract OCR, in which you can recognize any text, it is also possible to use it for other occasions that use a similar size and layout. In this case, I found, for example, a road sign used in Prague to identify parking spaces. This mark uses a one-line design with a specific symbol at the start. With a sufficient amount of data, an object detector model could be created to recognize just such a mark from a photograph.

Figure 5.12 Different use-case

In my case, I recognized this number plate by myself and used the rest of my implementation to identify the symbols. In Figure5.12 can be seen that all the symbols were recognized correctly, and therefore my implementation would be helpful even for other use-cases.

Chapter 6
Conclusion

The key objective of this thesis was to create an application that will recognize the text on the number plate for the most widely used number plate type: one line of Dark text with light background.

On the way to achieving the main objective, I have reviewed the literature dealing with automatic number plate recognition in images. After the implementation, I have tested my result on the preselected dataset and other use-cases.

The result is a Python application that will detect number plates with the help of YOLO version 4 implemented in TensorFlow and, as the next step, recognize text on the number plate using the open-source OCR Tesseract.

Using the OpenCV library, my implementation improved the results initially, using only Tesseract OCR and YOLO. The improvement in the correct recognitions was more than ten times better. I have also found more applicable use-cases where the implementation proved successful. It is admirable that such a result was achieved for a dataset containing different lighting conditions using only Tesseract OCR.

In the future, the application could be improved by not using Tesseract for OCR. Tesseract is mainly used for reading documents. Using other methods of character recognition, such as Convolutional Neural Network, could help improve the results. It would also be possible to improve the accuracy of the character localization. The ideal way would be to classify the number plate into different types that determine the exact position of the characters. This way, I would have the correct position of every symbol. This could improve the result up to the currently available services.

Appendix A
Results

First 30 results for EU dataset.

program_result	reference_result	is_program_result_empty	is_correct_detection	is_correct_length
M5XSX	M5XSX	false	true	true
WA56660	WA56660	false	true	true
IBS47049	BS47040	false	false	false
GWAGEN	GWAGEN	false	true	true
FESD	FWE50	false	false	false
BIMMIAN	BIMMIAN	false	true	true
OYO9FEUI	OYO9FEU	false	false	false
WOBVHIK4	WOBVWMK4	false	false	true
TWW4X4UP	VW4X4WP	false	false	false
HSO302	WSQ3021	false	false	false
	W053011	true	false	false
PP587A0	PP587AO	false	false	true
RK755AJI	RK755AJ	false	false	false
SI819AK	SI819AK	false	true	true
RK115AN	RK115AN	false	true	true
TS260AK	TS260AK	false	true	true
RKOSSAN	RKO99AN	false	false	true
ORK828AG	RK828AG	false	false	false
JLM298A	LM298AI	false	false	true
1T43213	1T43213	false	true	true
RK248AH	RK248AH	false	true	true
RK346AL	RK346AL	false	true	true
RK291AT	RK291AT	false	true	true
RK857A1	RK857AI	false	false	true
RKS76AHI	RK576AH	false	false	false
RKO19AF	RK019AF	false	false	true
BB751BH	BB751BH	false	true	true
RK867AD	RK867AD	false	true	true
RKBB4AL	RK884AL	false	false	true
PRKBE5AC	RK865AC	false	false	false

Bibliography

1. *Number plate*. Cambridge University Press, 2020. Available also from: https://dictionary.cambridge.org/dictionary/english/number-plate. [Online; Accessed 24-April-2022].
2. ALLO002. *File:license plate sizes.svg*. Wikimedia Foundation, 2012. Available also from: https://commons.wikimedia.org/wiki/File:License_plate_sizes.svg. [Online; Accessed 1-May-2022].
3. BASILLEAF. *File:Nepal License Plate - Private Car - Heavy Vehicle - 1983-2019.png*. Wikimedia Foundation, 2021. Available also from: https://commons.wikimedia.org/wiki/File:Nepal_License_Plate_-_Private_Car_-_Heavy_Vehicle_-_1983-2019.png. [Online; Accessed 1-May-2022].
4. SAJAD-HASANAHMADI. *File:.png*. Wikimedia Foundation, 2018. Available also from: https://commons.wikimedia.org/wiki/File:%D9%BE%D9%84%D8%A7%DA%A9_%D8%B4%D8%AE%D8%B5%DB%8C.png. [Online; Accessed 1-May-2022].
5. FARID, Nima. *File:Iraq - Kurdistan - License Plate - Private.png*. Wikimedia Foundation, 2020. Available also from: https://commons.wikimedia.org/wiki/File:Iraq_-_Kurdistan_-_License_Plate_-_Private.png. [Online; Accessed 1-May-2022].
6. WALSER.FL. *File:vehicle registration plates in Europe.png*. Wikimedia Foundation, 2020. Available also from: https://commons.wikimedia.org/wiki/File:Vehicle_registration_plates_in_Europe.png. [Online; Accessed 24-April-2022].
7. ČESKO. *VYHLÁŠKA ze dne 19. prosince 2014 o registraci vozidel*. 2014. Available also from: https://aplikace.mvcr.cz/sbirka-zakonu/ViewFile.aspx?type=z&id=27609. [Accessed 30-April-2022].
8. PUBLICATIONS OFFICE OF THE EUROPEAN UNION. *Rear registration plates on motor vehicles*. N-Lex, 2019. Available also from: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=LEGISSUM%3Ami0064. [Online; Accessed 24-April-2022].
9. ARTIFICIAL SOLUTIONS. *Homage to John McCarthy, the father of Artificial Intelligence (AI)*. Artificial Solutions, 2022. Available also from: https://www.artificial-solutions.com/blog/homage-to-john-mccarthy-the-father-of-artificial-intelligence#:~:text=McCarthy%20presented%20his%20definition%20of,AI%20research%20for%20many%20decades. [Online; Accessed 7-May-2022].
10. YAKOOVE. *File:fig-X all ML as a subfield of AI.jpg*. Wikimedia Foundation, 2020. Available also from: https://en.wikipedia.org/wiki/File:Fig-X_All_ML_as_a_subfield_of_AI.jpg. [Online; Accessed 24-April-2022].

11. MCCARTHY, John. *WHAT IS ARTIFICIAL INTELLIGENCE?* N. Alberto Borghese, [n.d.]. Available also from: https://borghese.di.unimi.it/Teaching/AdvancedIntelligentSystems/Old/IntelligentSystems_2008_2009/Old/IntelligentSystems_2005_2006/Documents/Symbolic/04_McCarthy_whatisai.pdf. [Online; Accessed 7-May-2022].
12. IBM CLOUD EDUCATION. *What is machine learning?* IBM, 2020. Available also from: https://www.ibm.com/cloud/learn/machine-learning. [Online; Accessed 7-May-2022].
13. IBM CLOUD EDUCATION. *What is Neural Networks?* IBM, 2020. Available also from: https://www.ibm.com/cloud/learn/neural-networks. [Online; Accessed 7-May-2022].
14. MATHWORKS. *What is deep learning?* MathWorks, [n.d.]. Available also from: %7Bhttps://www.mathworks.com/discovery/deep-learning.html#:~:text=Deep%20learning%20is%20a%20machine,a%20pedestrian%20from%20a%20lamppost.%7D. [Online; Accessed 7-May-2022].
15. IBM CLOUD EDUCATION. *What is computer vision?* IBM, [n.d.]. Available also from: https://www.ibm.com/cloud/learn/neural-networks. [Online; Accessed 7-May-2022].
16. *Computer vision market size, share report, 2021-2028*. Grand View Research, 2021. Available also from: %7Bhttps://www.grandviewresearch.com/industry-analysis/computer-vision-market#:~:text=Report%20Overview,7.3%25%20from%202021%20to%202028%7D. [Online; Accessed 7-May-2022].
17. ZHANG, Xinhua. *Gaussian distribution*. Springer, Boston, MA, 1970. Available also from: https://link.springer.com/referenceworkentry/10.1007/978-0-387-30164-8_323. [Online; Accessed 3-May-2022].
18. CHERNENKO, Sergey. *Gaussian filter, or Gaussian Blur*. Librow, [n.d.]. Available also from: http://www.librow.com/articles/article-9. [Online; Accessed 3-May-2022].
19. CHATTERJEE, Marina. *What is Computer Vision?* Great Learning, 2022. Available also from: https://www.mygreatlearning.com/blog/what-is-computer-vision-the-basics/. [Online; Accessed 1-May-2022].
20. REDMON, Joseph; DIVVALA, Santosh; GIRSHICK, Ross; FARHADI, Ali. *You Only Look Once: Unified, Real-Time Object Detection*. arXiv, 2015. Available from DOI: 10.48550/ARXIV.1506.02640. [Online; Accessed 26-April-2022].
21. REDMON, Joseph; FARHADI, Ali. *YOLO9000: Better, Faster, Stronger*. arXiv, 2016. Available from DOI: 10.48550/ARXIV.1612.08242. [Online; Accessed 26-April-2022].
22. REDMON, Joseph; FARHADI, Ali. *YOLOv3: An Incremental Improvement*. arXiv, 2018. Available from DOI: 10.48550/ARXIV.1804.02767. [Online; Accessed 26-April-2022].
23. BOCHKOVSKIY, Alexey; WANG, Chien-Yao; LIAO, Hong-Yuan Mark. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. arXiv, 2020. Available from DOI: 10.48550/ARXIV.2004.10934. [Online; Accessed 26-April-2022].
24. TENSORFLOW. *Tensorflow/tensorflow: An open source machine learning framework for everyone*. GitHub, Inc, [n.d.]. Available also from: https://github.com/tensorflow/tensorflow. [Online; Accessed 11-May-2022].
25. *About*. OpenCV, 2020. Available also from: https://opencv.org/about/. [Online; Accessed 11-May-2022].
26. *Languages supported in different versions of Tesseract*. tesseract-ocr, [n.d.]. Available also from: https://tesseract-ocr.github.io/tessdoc/Data-Files-in-different-versions.html. [Online; Accessed 28-April-2022].
27. SMITH, R. An Overview of the Tesseract OCR Engine. In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. 2007, vol. 2, pp. 629–633. Available from DOI: 10.1109/ICDAR.2007.4376991.

28. *Improving the quality of the output*. [N.d.]. Available also from: https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html. [Online; Accessed 11-May-2022].

29. LEE, Matthias. *Pytesseract*. Python Software Foundation, [n.d.]. Available also from: https://pypi.org/project/pytesseract/. [Online; Accessed 10-May-2022].

30. NIXON, Mark S.; AGUADO, Alberto S. *Feature extraction and image processing*. Newnes, 2006. ISBN 0750650788. Available also from: https://theswissbay.ch/pdf/Gentoomen%20Library/Artificial%20Intelligence/Computer%20Vision/Feature%20Extraction%20in%20Computer%20Vision%20and%20Image%20Processing%20-%20Mark%20S.%20Nixon.pdf. [Accessed 1-May-2022].

31. *Gaussian blur*. Wikimedia Foundation, 2022. Available also from: https://en.wikipedia.org/wiki/Gaussian_blur. [Online; Accessed 27-April-2022].

32. ADOBE. *Using gaussian blur in image processing | adobe*. Adobe, [n.d.]. Available also from: https://www.adobe.com/creativecloud/photography/discover/gaussian-blur.html. [Online; Accessed 1-May-2022].

33. OPENCV. *GaussianBlur*. OpenCV, [n.d.]. Available also from: https://docs.opencv.org/4.x/d4/d86/group__imgproc__filter.html#gaabe8c836e97159a9193fb0b11ac52cf1. [Online; Accessed 3-May-2022].

34. IKAMUSUMEFAN. *File:Cappadocia Gaussian Blur.svg*. Wikimedia Foundation, 2015. Available also from: https://commons.wikimedia.org/wiki/File:Cappadocia_Gaussian_Blur.svg. [Online; Accessed 27-April-2022].

35. ELKOSOLTIUS. *File:Example of adaptive thresholding.png*. Wikimedia Foundation, 2022. Available also from: https://commons.wikimedia.org/wiki/File:Example_of_adaptive_thresholding.png. [Online; Accessed 24-April-2022].

36. *Contour approximation method*. [N.d.]. Available also from: https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html. [Online; Accessed 11-May-2022].

37. SUZUKI, Satoshi; BE, KeiichiA. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*. 1985, vol. 30, no. 1, pp. 32–46. ISSN 0734-189X. Available from DOI: https://doi.org/10.1016/0734-189X(85)90016-7.

38. HUNGLC007. *HUNGLC007/tensorflow-yolov4-tflite: Yolov4, Yolov4-tiny, yolov3, yolov3-tiny implemented in tensorflow 2.0, Android. Convert Yolo V4 .Weights tensorflow, tensorrt and tflite*. [N.d.]. Available also from: https://github.com/hunglc007/tensorflow-yolov4-tflite. [Online; Accessed 11-May-2022].

39. *Custom.weights*. Google, [n.d.]. Available also from: https://drive.google.com/file/d/1EUPtbtdF0bjRtNjGv436vDY28EN5DXDH/view?usp=sharing. [Online; Accessed 11-May-2022].

40. OPENALPR. *Benchmarks/endtoend/EU at MASTER · openalpr/benchmarks*. [N.d.]. Available also from: https://github.com/openalpr/benchmarks/tree/master/endtoend/eu. [Online; Accessed 11-May-2022].

Contents of the media

- impl
  - conda-cpu.yml ..... environment python environment
  - conda-gpu.yml ..... environment python environment
  - detect.py ..... init part of the implementation
  - README.md ..... user guide
  - requirements-gpu.txt ..... environment python environment
  - requirements.txt ..... environment python environment
  - save_model.py ..... implementation file
  - core
    - backbone.py ..... implementation file
    - common.py ..... implementation file
    - config.py ..... implementation file
    - dataset.py ..... implementation file
    - plate_recognition.py ..... main file of my implementation
    - utils.py ..... implementation file
    - yolov4.py ..... implementation file
  - data
    - anchors
      - yolov4_anchors.txt ..... implementation file
    - classes
      - custom.names ..... implementation file
    - example
      - eu1.jpg ..... example file for implementation
      - eu1.txt ..... example file for implementation
- thesis
  - ctufit-thesis.pdf ..... final form of thesis
  - src
    - assignment-include.pdf ..... assignment
    - ctufit-thesis.cls ..... thesis file for LaTeX
    - ctufit-thesis.tex ..... thesis file for LaTeX
    - images
      - 1_locate.png ..... image used in thesis
      - 2_gray.png ..... image used in thesis
      - 3_blur.png ..... image used in thesis
      - 4_binarized.png ..... image used in thesis
      - 5_all_contours_binarized.png ..... image used in thesis
      - 5_all_contours_color.png ..... image used in thesis

- 6_filtered_conts_binarized.png ..... image used in thesis
- 6_filtered_conts_color.png ..... image used in thesis
- 7_chars_on_white_background.png ..... image used in thesis
- 800px-Cappadocia_Gaussian_Blur.png ..... image used in thesis
- 8_final.png ..... image used in thesis
- AI_subfield.jpg ..... image used in thesis
- alphanumeric_CZ.png ..... image used in thesis
- ANN_diagram.png ..... image used in thesis
- cz_sizes.png ..... image used in thesis
- different_usecase.png ..... image used in thesis
- Example_of_adaptive_thresholding.png ..... image used in thesis
- gaussian_function.png ..... image used in thesis
- gauss_2d.png ..... image used in thesis
- Iran_plate.png ..... image used in thesis
- Iraq_plate.png ..... image used in thesis
- License_plate_sizes.png ..... image used in thesis
- Nepal_License_Plate.png ..... image used in thesis
- program_flow.png ..... image used in thesis
- registration_plates_Europe.png ..... image used in thesis
- Russian_license_plate.png ..... image used in thesis
- white_black_numbers.png ..... image used in thesis
- white_black_paid.png ..... image used in thesis
- white_black_stars.png ..... image used in thesis
- white_blue.png ..... image used in thesis
- white_green_long.png ..... image used in thesis
- white_green_short.png ..... image used in thesis
- yellow_black.png ..... image used in thesis
- yolo.png ..... image used in thesis
- YOLOv4.png ..... image used in thesis

text

- appendix.tex ..... thesis file for LaTeX
- bib-database.bib ..... thesis file for LaTeX
- medium.tex ..... thesis file for LaTeX
- text.tex ..... thesis file for LaTeX

tstex_modules

- _api.ts ..... thesis file for LaTeX