BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

TRAFFIC SURVEILLANCE SYSTEM

SYSTÉM PRO SLEDOVÁNÍ DOPRAVY

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **JAKUB VAŇO**
AUTOR PRÁCE

SUPERVISOR **Ing. ROMAN JURÁNEK, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2024

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Bachelor's Thesis Assignment

157986

Institut:	Department of Computer Graphics and Multimedia (DCGM)
Student:	**Vaňo Jakub**
Programme:	Information Technology
Title:	**Traffic surveillance system**
Category:	Computer vision
Academic year:	2023/24

Assignment:

1. Study methods for traffic surveillance. Focus on detection, tracking, recognition and speed estimation of objects typical for traffic scenes (vehicles, persons).
2. Design your real-time system enabling long-term surveillance and analysis of captured data.
3. Implement the designed system. If required, train models for detection, tracking and recognition on existing datasets.
4. Develop front-end information system which will enable users to manage, view and analyze data from the surveillance.
5. Deploy your system for long-term video surveillance and analyze captured data.
6. Create materials presenting your work

Literature:

1. GREGOR, Adam. Vehicle Make and Model Recognition. Brno, 2023. Master's thesis. Brno University of Technology, Faculty of Information Technology
2. Ultralytics, https://github.com/ultralytics/ultralytics
3. Kerberos.io, https://kerberos.io/

Requirements for the semestral defence:
Items 1 and 2.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Juránek Roman, Ing., Ph.D.**
Head of Department:	Černocký Jan, prof. Dr. Ing.
Beginning of work:	1.11.2023
Submission deadline:	9.5.2024
Approval date:	9.11.2023

Abstract

This thesis concerns with different methods for traffic surveillance systems and data gathering tied with it. Different object detection and tracking methods with main focus on Neural Networks were studied, among which YOLO has been chosen and used for implementation of vehicle detection. The final implementation of traffic surveillance system contained the camera worker, task manager and three different scripts for data enrichment. System communication was achieved with MQTT broker. The results of short system running prove its capability to withstand real-time data gathering and processing from multiple cameras. Results of gathered data were analyzed with use of Grafana. After information gathering for sufficient amount of time, these data could potentially be used for traffic improvements.

Abstrakt

Táto práca sa zaoberá rôznymi metódami pre systémy sledovania dopravy a zberom dát s tým spojeným. Boli naštudované rôzne metódy detekcie a sledovania objektov so zameraním na neurónové siete, spomedzi ktorých bol vybraný a použitý model YOLO na implementáciu detekcie vozidiel. Konečná implementácia systému sledovania dopravy obsahovala správcu úloh a tri rôzne programy na doplnenie dát. Komunikácia systému bola dosiahnutá pomocou MQTT agenta. Výsledky krátkodobého testovania systému potvrdzujú jeho schopnosť zvládnuť zber a spracovanie dát v reálnom čase z viacerých kamier. Výsledky zozbieraných dát boli analyzované pomocou Grafany. Po dostatočnom čase zberu informácií by tieto údaje mohli byť potenciálne použité na zlepšenie dopravy.

Keywords

traffic surveillance, vehicle detection, vehicle tracking, vehicle classification, YOLOv8, OpenCV CLIP

Klíčová slova

sledovanie dopravy, detekcia vozidiel, sledovanie vozidiel, klasifikácia vozidiel, YOLOv8, OpenCV CLIP

Reference

VAŇO, Jakub. *Traffic surveillance system.* Brno, 2024. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Roman Juránek, Ph.D.

Rozšířený abstrakt

Úvod

Sledovanie dopravy je dôležitou súčasťou každého moderného mesta, ktoré sa snaží pochopiť, kontrolovať a efektívne upravovať svoje cesty, aby čo najlepšie slúžili obyvateľom. Sledovanie dopravy má mnoho praktických využití, ako je napríklad sledovanie plynulosti premávky, ktoré dokáže poukázať na problematické úseky cesty či križovatky. Ďalším praktickým využitím je kontrola dodržiavania rýchlosti a iných cestných zákonov. Vďaka technologickým pokrokom vie byť strojové učenie použité na efektívne sledovanie dopravy.

Cieľom tejto práce je vytvoriť systém pre sledovanie dopravy, ktorý dokáže zbierať dáta z viacerých kamier v reálnom čase. Zároveň sú tieto dáta obohatené a uložené na analýzu, z ktorej sa dajú vyvodiť užitočné štatistiky o doprave.

Popis riešenia

Konečná implementácia môže byť rozdelená na dve časti. Prvou je program obslujúci kameru, ktorý v reálnom čase detekuje a sleduje vozidlá s využitím neurónovej siete YOLOv8. Tento program zbiera základné informácie o detekovanom vozidle, ktoré sú dočasne uložené v Redis databáze a následne oznamuje novú detekciu zvyšku systému na dodatočné spracovanie. Druhú časť systému tvorí správca úloh, ktorý preberá zachytené vozidlá a rozposiela úlohy na doplnenie dát ďalším programom, ktorý sa špecializujú na počítanie rýchlosti, rozpoznanie cestných podmienok a klasifikáciu vozidla. Konečné dáta správca úloh uloží do PostgreSQL databázy, z ktorej ich Grafana spracuje na súbor štatistík. Komunikáciu v systéme zabezpečuje MQTT agent. Na zistenie cestných podmienok bola použitý OpenAI CLIP, ktorý dokáže detekovať objekty v obraze na základe kľúčových slov. Prebraný natrénovaný model neurónovej siete ResNet152 na ropoznanie značky a modelu vozidiel bol použitý pre klasifikáciu vozidiel spolu s OpenAI CLIP pre zistenie farby vozidla.

Zhrnutie výsledkov a ďalšia práca

Výsledky krátkodobého testovania systému dokazujú jeho schopnosť zbierať a spracúvavať dáta v reálnom čase z viacerých kamier. Systém úspešne detekoval vozidlá, odhadoval ich rýchlosť, klasifikoval ich na základe značky, modelu a farby a správne rozpoznával cestné podmienky. Analýza dát pomocou Grafany umožnila vytvoriť rôzne grafy a tabuľky, ktoré poskytujú prehľad o dopravnej situácii. Pre lepšie pochopenie dopravy by bolo potrebné dlhodobé sledovanie. Možné vylepšenia systému sú natrénovanie vlastného modelu na klasifikáciu vozidiel na základe lokálnych dát a získavanie ďalších informácií ako sú napríklad evidenčné čísla vozidiel.

Traffic surveillance system

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Ing. Roman Juránek, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Jakub Vaňo
May 15, 2024

Acknowledgements

Many thanks to my work supervisor who helped this work come to fruition. Thank you Ing. Roman Juránek, Ph.D. for your support, valuable advice, time and guidance.

Contents

Chapter 1

Introduction

Traffic surveillance is important for every modern city that aims to understand, control and adapt its road infrastructure to suite the traffic flow it encounters. Thanks to technology advances in the recent years, machine learning can be used to effectively and efficiently study traffic. Camera is the most commonly used method for traffic surveillance as it allows for vehicle detection, tracking and classification using neural networks. With this information statistics for road usage can be created that help find the problems such as road bottlenecks or frequent speeding and reckless driving areas.

This thesis aims to create a system capable of detecting vehicles with camera. Processing detected vehicles with neural networks to extract such information as manufacturer, model and colour. Speed estimation can be done as well using information about the camera and trajectory of detected vehicle. Other useful information is road condition that can be estimated from scene image. These data can be then used to create meaningful statistics about traffic on the road that the camera oversees. For this, understanding machine learning is essential as it is needed for implementation of detection and subsequently classification of vehicles.

The implemented system is capable of real time surveillance from multiple cameras. Resulting database of detected vehicles is used to draw various traffic related graphs and information tables. This thesis uses MQTT for communication between parts of the system, Redis as a cache memory before final data are stored in PostgreSQL and Grafana for analysis of final data and its visual representation.

Chapter 2 introduces various options of gathering data about traffic and how can be the data used. Chapter 3 talks about different ways of object detection and tracking. Chapter 4 describes object classification methods and OpenAI CLIP. Chapter 5 focuses on implementation details and Chapter 6 presents the results and their implication.

Chapter 2

Traffic Data Gathering

Over the years multiple methods of gathering useful information about traffic has been used. The most simple option to gather information about traffic is by human observation. Basic information such as number of vehicles and their direction can be gathered this way. However its inefficiency and possibility of human errors make it inappropriate for large modern cities. The most well known modern method of traffic observation is camera. Other useful visual methods include the use of LiDAR, infrared cameras and road sensors like load sensors, or magnetic field sensors. In this chapter, practical uses for traffic surveillance are discussed, after that, some of the methods to gather information about traffic are explored.

2.1 Practical Uses of Gathered Information

All methods of gathering information about traffic, discussed in next section, collect crucial data that are used in countless real-life applications. Technology advancements allow for deployment of city wide traffic surveillance systems for research purposes, as well as commercial and safety uses. Some of the most common uses are traffic flow management, urban planning, and development usually conducted by the city. Emergency services make use of traffic surveillance to improve safety and law enforcement. Other then that it is also applied in various private sector commercial uses.

Traffic Flow Management is important for large cities as the road congestion can make life challenging. Traffic light intervals can be improved by observing traffic density in real-time. City wide traffic surveillance capable of tracking vehicles between cameras can help understand how people get around the city. Example of intersection surveillance is shown in Figure 2.1.

Law Enforcement and Safety Enhancement can save lives as continuous traffic surveillance helps quickly detect accidents for faster emergency responses. Another way of improving safety on road is by police enforcing speed limits and other traffic laws with autonomous surveillance systems. Pedestrian safety at crosswalks and sidewalks can be improved by monitoring and managing interactions between pedestrians and vehicles.

Urban Planning and Infrastructure Development is another sector where cities can profit from the use of modern surveillance system. By analyzing long-term traffic data, future plans for road designs and modifications can be improved. Combining traffic surveillance with environmental sensors to monitor pollution levels and noise can influence regulatory actions. Public transport can benefit from traffic data by optimizing bus routes, schedules as well as enforcing proper use of dedicated lanes.

Commercial Use of traffic data can help optimize store locations and advertising strategies. Car manufacturers benefit from information like vehicle type and colour popularity that are easily accessible with traffic surveillance. Control gates and highway toll systems can be automated by employing surveillance methods.

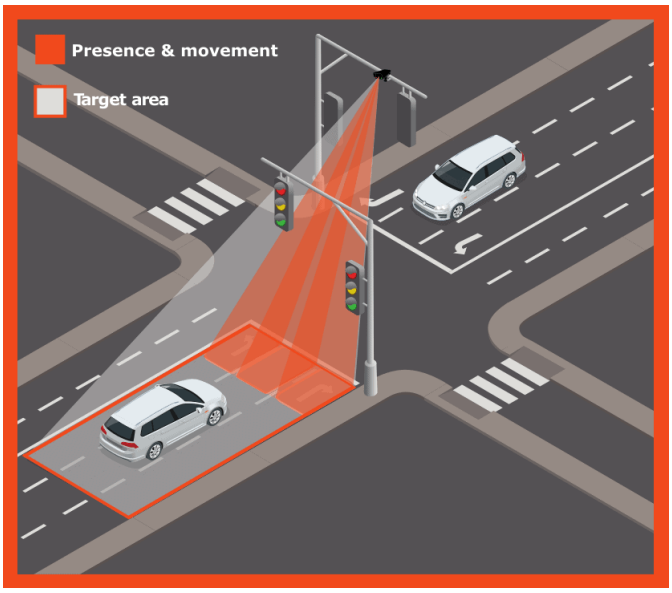

Figure 2.1: **Example of camera used for intersection surveillance** Source: [8]

Other additional information like license plate numbers can be used by the police to find a stolen vehicle or by private companies to track vehicles in their parking lots.

2.2 Data gathering methods

As Shokravi *et al.* [21] discussed, traffic data gathering methods can be divided into intrusive and non-intrusive methods. Both categories are capable of gathering useful information that can help improve traffic in urban areas.

Intrusive Methods

Intrusive methods collect data by using various sensors placed directly on road. These sensors can be load and vibration, or magnetic.

Magnetic sensors are capable of detecting distortion in the Earth's magnetic field caused by passing vehicle. Magnetic loop sensors are the most common type of magnetic sensors used in traffic surveillance. Simple visualization of magnetic loop sensor is shown in Figure 2.2.

Single-loop detectors capture the vehicle as it passes over the loop. Extension of simple single-loop detector is a dual-loop detector where two sensors are positioned few meters apart, providing more detailed data like speed estimation.

Major advantage for magnetic sensors is their relative simplicity and low cost. They also perform well under various weather conditions. Disadvantages are intrusive and long installation, together with need of proper calibration [21].

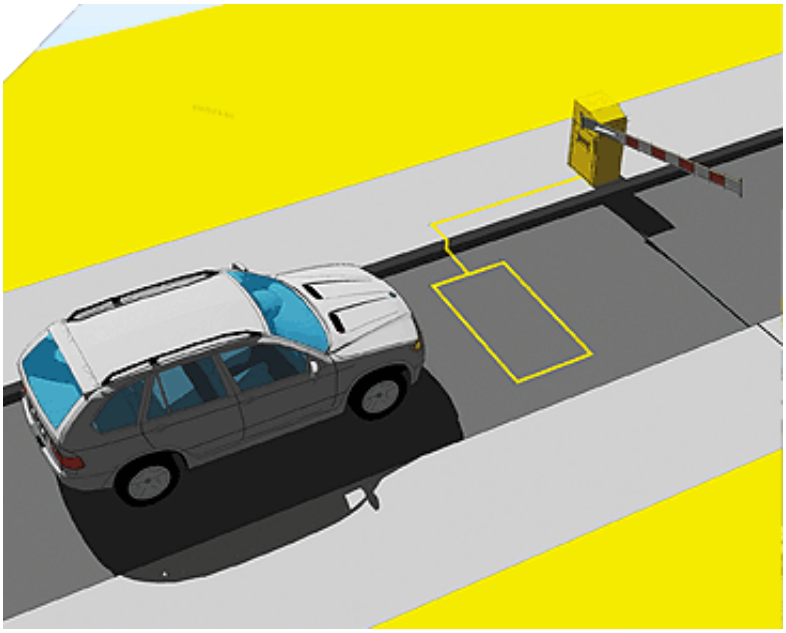

Figure 2.2: **Single-loop magnetic sensor used for control gate access** Source: [20]

Second category of intrusive sensors include Pneumatic Tubes that were first used in 1920 and are primarily used for temporary traffic observation. They work by being laid across road and register air pressure change caused by vehicle passing over them. Other then counting vehicles, number of vehicle axles can be detected, which can be used to estimate speed. The pressure created by passing vehicle can be used for weight estimation.

Piezoelectric sensors convert mechanical pressure into electrical signals. They are embedded below the pavement and are capable of vehicle counting and axle spacing estimation. In addition if the same vehicle triggers two sensors, vehicle speed can be estimated.

Strain Gauge sensors are embedded directly into the road, and they measure the deformation caused by vehicles. This method is useful for identifying different types of vehicles based on the unique strain signatures they produce.

Lastly, Seismic sensors detect vibrations in the ground caused by passing vehicles. Identification of vehicle type is possible by analyzing the pattern of vibrations [21].

Non-intrusive Methods

Camera is the best known visual-based non-intrusive method. It is widely studied and offers countless methods of detection, tracking and classification. Camera allows gathering wide range of data about traffic and offers use in multiple surveillance systems. For example it can be used for close up vehicle detection where license plate information or driver picture are important.

LiDAR is a remote sensing technology capable of generating Doppler signals for the detection of distributed or solid targets. LiDAR functions by transmitting and receiving electromagnetic radiation, and the extracted characteristics from vehicles are analyzed after the processing of the data. Main advantages of LiDAR are its high resolution and precision compared to radar, however it is more expensive and does not perform as well in rain and snow.

Infrared cameras also known as thermal work well at night, but are expensive. They are commonly used for the detection of vehicles in the battlefield [21].

Figure 2.3: **Example of LiDAR camera used for smart traffic management** Source: [23]

Chapter 3

Object Detection and Tracking

Traffic surveillance plays a major role in collecting critical data that are used to enhance traffic flow and safety. Thanks to technology progress in recent years, the process of traffic monitoring, its analysis and management has changed greatly. This chapter introduces technologies that are essential for traffic surveillance, starting with object detection methods, deep learning applications and object tracking technologies.

The main purpose of object detection is to identify every occurrence of objects from one or multiple predefined classes, such as people, cars, or faces, within an image. Every detection comes with some form of information. These information can be location, object bounding box, or segmentation mask. In some cases, the information is more comprehensive and includes parameters for a linear or nonlinear transformation. Example of such information can be location of the headlights, and license plate, together with bounding box of a car. Example of bounding box information from multiple detected objects is shown in Figure 3.1.

Object detection systems create models for object classes using training examples, a single example may be enough for rigid object, but capturing class variability usually require many more. Designing model to handle extensive variability in images can be challenging. Alternatively, convolutional neural networks (CNN) learn to handle this variability by training on large datasets [1].

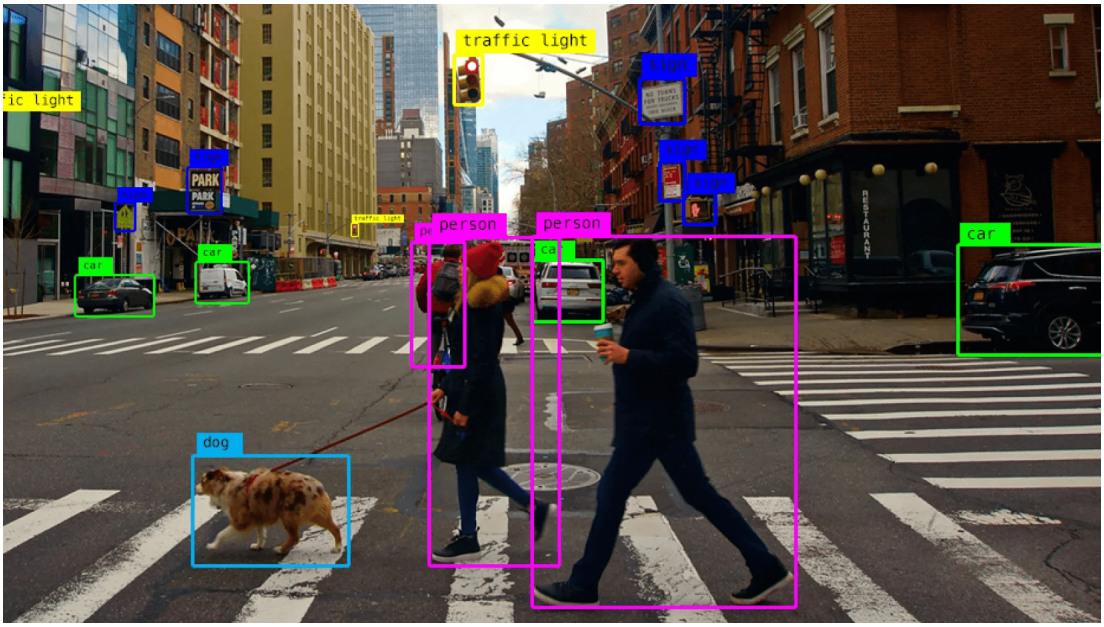

Figure 3.1: **Object detection with bounding box information** Source:[2]

3.1 Feature-Based Detection

Feature-based detection methods can be categorized into two types: global-feature based and local-feature based matching algorithms. Local-feature based matching algorithms, compared to the global-feature based, are more stable and have been used successfully in various practical applications. These include object and texture recognition, image retrieval, robot localization, and object category recognition.

Local feature-based matching algorithms are composed of two stages: interest point detection and description. Effective local features should have certain key characteristics: the feature detection process should be highly repeatable and fast, while the feature description should have a low feature dimension, that helps in rapid matching and strength to changes in illumination, rotation, and viewpoint change. SIFT is a local feature description algorithm proposed by David G. Lowe, it builds on existing invariance-based feature detection methods available at that time. SIFT is notable for its stability and invariance, detecting local keypoints rich in information. Due to its distinct benefits, SIFT has become widely studied topic, with many researchers dedicated to refining and enhancing the technique [24].

3.2 Neural Networks and Deep Learning

Neural Network is a machine learning method, utilizing layers of artificial neurons to learn from vast amount of data. Deep learning is an extension of neural networks, utilizing complex structures with multiple layers to extract high-level features from raw input automatically, offering significant improvements over traditional methods that require manual feature selection.

As explained by O'Shea *et al.*[14] convolutional neural networks (CNN) are build on the foundation of standard neural networks, with the difference of having three types of

hidden layers. Example of such architecture is shown in Figure 3.2 These are convolutional, pooling, and fully-connected layers. These layers can be described as such:

- The convolutional layer computes the output of neurons that are connected to specific local areas of the input by calculating the dot product between their weights and the region connected to the input volume. The rectified linear unit (ReLU) serves to implement an 'elementwise' activation function, like the sigmoid, to the output generated by the preceding layer.
- The pooling layer reduces complexity by performing down-sampling across the spatial dimension of the input, which decreases the number of parameters in that activation
- The fully-connected layers then carry out functions similar to those in standard Neural Networks, working to generate class scores from the activations for classification purposes. It is also recommended to use ReLu between these layers to enhance performance.

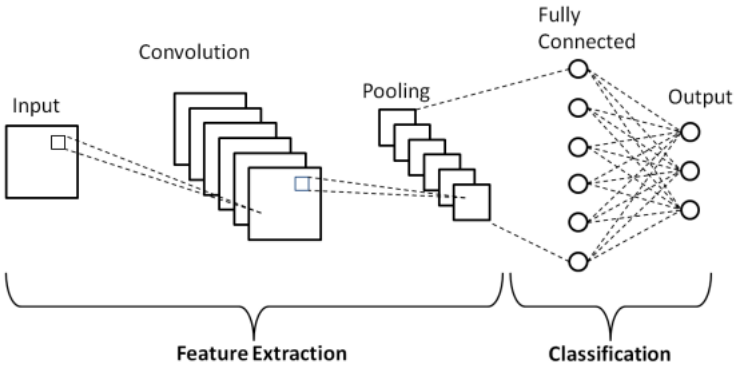

Figure 3.2: **Example of simple CNN architecture** Source:[15]

Region-Based Convolutional Neural Network (R-CNN)

R-CNN is a basic deep neural network that is designed for object detection tasks. It uses selective search to generate object proposals and train CNNs, producing 2000 candidate boxes. Each box is resized and inputted into the CNN, which extracts features and outputs a 4096 dimensional feature as a output. These features are classified using an SVM, and R-CNN also adjusts four offset values to refine the precision of each bounding box. System overview is shown in Figure 3.3

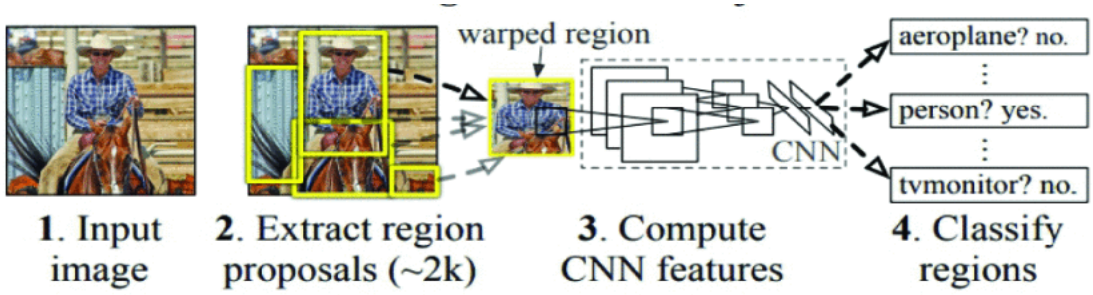

Figure 3.3: **Overview of R-CNN** Source:[12]

Main R-CNN limitations are:

- Substantial time complexity, that renders R-CNN impractical for real life application
- The generation of candidate region proposals is often inaccurate because the selective search algorithm lacks inherent learning capabilities

Fast R-CNN

Compared to regular R-CNN, Fast R-CNN does not need the 2000 region proposals generated by the selective search method as input. Instead, it inputs the entire image into the CNN to create a feature map. This map is then used to identify and resize region proposals to a uniform size using a RoI pooling layer. Subsequently, a softmax layer determines the objects within each proposal and predicts four offset values. Overview of Fast R-CNN can be see in Figure 3.4

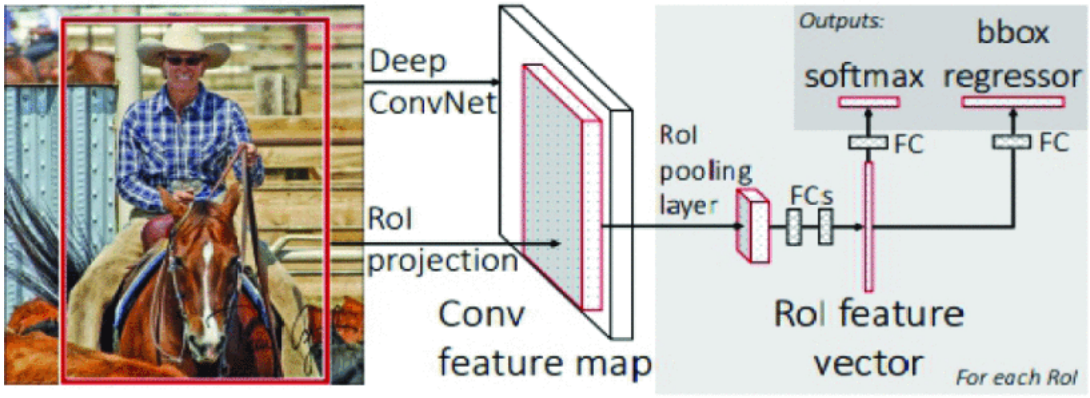

Figure 3.4: **Overview of Fast R-CNN** Source:[12]

Main Fast R-CNN limitation is that even with introduction of RoI pooling layer that reduces the time complexity compared to R-CNN, the problem of inaccurate region proposals generation persists.

Faster RCNN

In order to deal with limitations of R-CNN and Fast R-CNN, Faster R-CNN is created by combining Fast R-CNN with a new fully-convolutional network called the Region Proposal Network (RPN). The RPN not only produces high-quality region proposals, but also concurrently suggests object bounds and objectness scores at each location.

Faster R-CNN is created by combining Fast R-CNN with a new fully-convolutional network called the Region Proposal Network (RPN). The RPN not only produces high-quality region proposals but also concurrently suggests object boundaries and objectness scores at each location.

Thanks to better performance in accurate region proposal generation and further reduction in time-complexity compared to R-CNN and Fast R-CNN, Faster R-CNN is used widely by researchers. Overview of Faster R-CNN can be seen in Figure 3.5

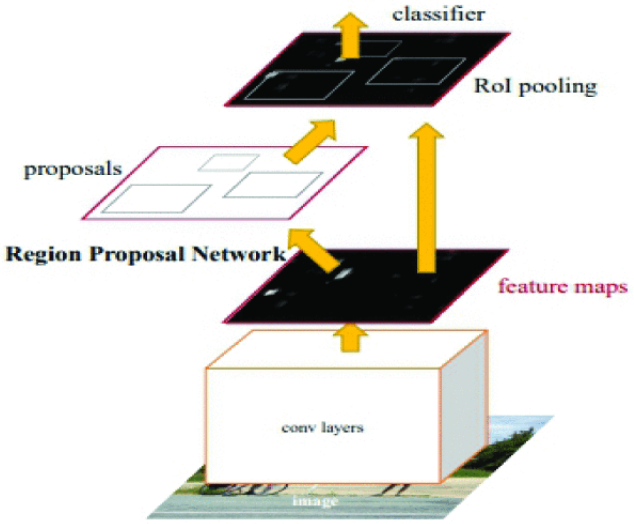

Figure 3.5: **Overview of Faster R-CNN** Source:[12]

YOLO (You Only Look Once)

YOLO has been created by Redmon *et al.*[18] to reduce run-time complexities of R-CNN and its variants. Instead of requiring region proposals to localize and classify objects, YOLO models take whole image and split it into $S \times S$ grid. For each grid YOLO locates 'm' number of bounding boxes. Each bounding box predicts a class and offset values. If the predicted class probabilities are below a specific threshold, the bounding box is suppressed [12]. Example of YOLO workflow is shown in Figure 3.6.

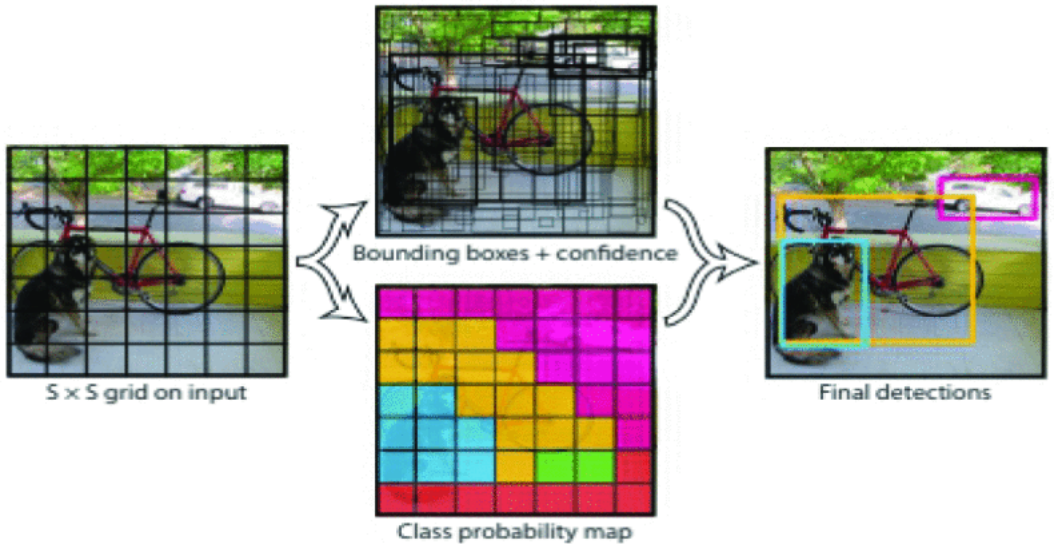

Figure 3.6: **Example of YOLO workflow** Source:[12]

As explained by Huang *et al.* [6] confidence that an object is present within each bounding box is defined as:

$$C = \Pr(Object) * IOU_{pred}^{truth} \tag{3.1}$$

where IOU, or intersection over union, is a value ranging from 0 to 1. It measures the overlap between the predicted bounding box and the actual ground truth. IOU close to 1 indicates a high degree of accuracy, meaning the predicted bounding box closely matches the ground truth.

At the same time as the bounding boxes are being created, each grid cell also predicts C conditional class probabilities. The probability specific to each class for each grid cell is defined as follows:

$$\begin{aligned} &\Pr\left(Class_i|Object\right) * \Pr\left(Object\right) * IOU_{pred}^{truth} \\ &= \Pr\left(Class_i\right) * IOU_{pred}^{truth}. \end{aligned} \tag{3.2}$$

The loss function adjusts the center and dimensions of each bounding box in prediction. To calculate loss and enhance confidence, YOLO uses multi-part loss function during training. It calculates errors in bounding box coordinate prediction, confidence score prediction, and class prediction. [6].

While loss measures the effectiveness of a model, the accuracy of prediction in object detection is determined using the average precision equation, which is defined as:

$$avgPrecision = \sum_{k=1}^{n} P\left(k\right) \Delta r\left(k\right). \tag{3.3}$$

where $P(k)$ represents the precision at the threshold k while $\Delta r(k)$ indicate the change in recall.

The architectural design of the YOLO neural network includes 24 convolutional layers followed by 2 fully connected layers. This architecture is shown in Figure 3.7.

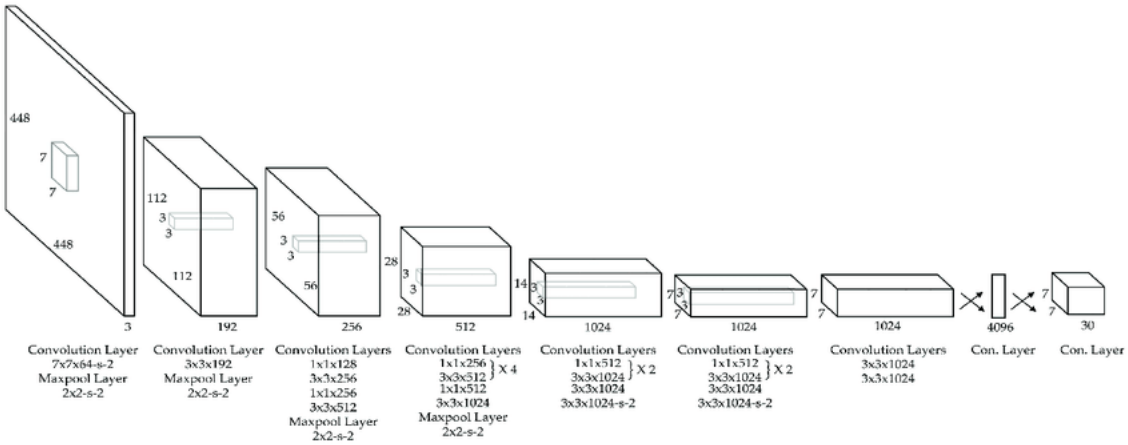

Figure 3.7: **YOLO architecture** Source:[9]

YOLO has come through a number of versions over the years and as Hussain [7] discussed, notable changes in development are:

- **YOLO-v2** Training speed was increased with introduction of batch normalization, and instead of using fully connected layers for bounding box prediction, anchor boxes were used.
- **YOLO-v3** Focused on improving detection of smaller objects by doubling the number of convolutional layers, as well as making prediction at three different scales of granularity, increasing chances of small object detection.

- **YOLO-v4** Continued to improve performance by implementing mosaic data augmentation, Mish activation function and Cross mini-batch normalization.
- **YOLO-v5** Was a first version written in PyTorch which made it far more accessible. In addition automated anchor box learning was introduced.
- **YOLO-v6 and YOLO-v7** Released just one month apart and both focused on further improving training speed and detection efficiency. YOLO-v6 notably decided for an anchor-free approach and YOLO-v7 made architectural reforms such as implementation of E-ELAN (Extended Efficient Layer Aggregation Network).

YOLO-v8 Is the latest version of YOLO developed by Ultralytics that also developed YOLO-v5. YOLO-v8 uses the same architecture as their previous version, but it introduces several enhancements over previous version. The architecture is shown in Figure 3.8. It introduces a new neural network architecture that uses Feature Pyramid Network (FPN) as well as Path Aggregation Network (PAN). Additionally, it features a new labeling tool designed to simplify the annotation process. This tool includes several practical features such as automatic labeling, labeling shortcuts, and customisable hotkey. These make it easier to annotate images for training the model.

The FPN works by progressively decreasing the spatial resolution of the input image while increasing the number of feature channels. This process generates feature maps that can detect objects across various scales and resolutions. The PAN aggregates features from various network layers using skip connections. This approach enables to effectively capture features at multiple scales and resolutions, which is essential for the precise detection of objects varying in size and shape [19].

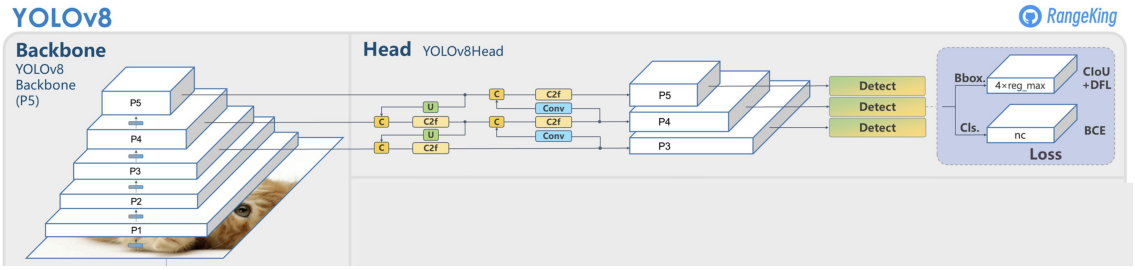

Figure 3.8: **Overview of YOLO-v8 architecture**. Source: [19]

3.3 Object Tracking

As discussed by Yilmaz *et al.* [26] tracking the detected object is essential for any surveillance system. The purpose of an object tracker is to create the trajectory of an object through time by pinpointing its location in each frame of the video. Object tracker can also provide the entire area in the image where the object is located at each moment. The tasks of detecting the object and linking instances of the object across different frames can be done either independently or together. In the independent method, object regions are identified in each frame using an object detection algorithm, and then the tracker matches these objects across frames. In the integrated method, both the object region and its correspondence are determined together by iteratively updating object location and region data from previous frame.

Object tracking methods can be divided into multiple categories such as, point tracking, kernel tracking, silhouette tracking, and multi-object tracking.

Point Tracking involves matching detected objects, represented as points, across video frames. Point tracking is particularly challenging due to issues like occlusions, misdetections, and the entry or exit of objects from the scene. This method can be categorized into two types: deterministic and statistical methods. Deterministic methods apply qualitative motion heuristics to limit the challenges of tracking, whereas statistical methods account for object measurement and uncertainties to determine tracking.

Kernel Tracking usually involves analyzing the motion of an object, depicted by a basic object region, from one video frame to the next. This motion is typically represented through parametric motions (such as translation, conformal, affine) or through dense flow fields calculated in subsequent frames. Kernel tracking can be divided into two categories: templates and density-based appearance models, and multiview appearance models.

Silhouette Tracking provide a precise description of complex shapes. Objects with intricate shapes, such as hands, heads, and shoulders, often cannot be accurately represented using simple geometric shapes. The objective of silhouette based object tracker is to identify the objects region in each frame using an object model constructed from previous frames. This model can be in a form of color histogram, object edges or the object contour. These trackers can be categorized into two main types: shape matching and contour tracking. Shape matching methods search for the object silhouette in the current frame. Contour tracking methods adapt an initial contour to its updated position in the current frame through state space models or by directly minimizing an energy functional.

For this thesis ByteTrack is used for tracking which is a **Multi-Object Tracking** (MOT) model. It is a real-time object tracking algorithm. MOT contains two stages. The initial stage involves object detection, which identifies the position and categories of relevant classes within video frames and assigns a unique instance ID to each object. Object detection methods like R-CNN or YOLO is used. The second stage, known as association stage, utilizes temporal data from various frames to create a trajectory for each object [10].

Chapter 4

Object Classification

To enhance traffic surveillance further, various added information can be obtained about detected vehicles. This information cam help identify vehicles detected in multiple cameras, which in turn can be used to determine path of the vehicle through a city.

Technology advancements make large scale complex traffic surveillance systems possible. Instead of using current surveillance methods only to count passing vehicles, various useful information can be obtained by further data analysis. Some of the most useful information is the type of vehicle (car, truck, bus), license plate number, vehicle brand and model, and its colour. In order to obtain this information, variety of methods can be used. The most simple would be to classify detected vehicles by hand. This option however is highly time consuming and impractical in modern world.

Support Vector Machines (SVM) is very effective machine learning algorithm for classification and pattern recognition. Usually used for binary classification, it is possible to adapt for multi class classification. SVM creates a hyperplane dividing points of the same class to one side. [4]. In this manner vehicle types can be categorized based on features like shape, and size.

Bayesian Network Classifiers are a type of Bayesian networks created for classification tasks. They assign labels or categories to instances based on a set of features, using a model that is automatically generated from labeled data.

Neural Networks are one of the best option for object classification. Described in Chapter 3 CNNs are especially good at object classification. Some notable CNN models are: ResNet, Xception, and VGG16. Neural networks require large amount of labeled image data for training.

4.1 Available Datasets

As Gregor [5] discussed, datasets are essential for training of neural networks. Some of the current vehicle datasets are shown in this section. The main disadvantage of these datasets is the fact they focus only on certain country.

Stanford Cars Dataset Created by Stanford University in 2013, it contains 16 185 images of 196 classes of cars. This dataset is based on cars in USA. Data are classified as car's manufacturer, model name and production year. In addition to these classes, a bounding box is provided. Example of images from dataset is shown in Figure 4.1.

Figure 4.1: **Example images from Stanford Dataset** Source:[13]

CompCars Dataset is a dataset from China created in 2015. It is split into two parts, first is a web-based part containing 136 727 images from Internet from mostly Chinese car manufacturers. Second is surveillance-based part with 44 481 images of cars. As previous dataset cars are classified with car's manufacturer, model name and production year. In addition, maximum speed, displacement, number of doors, number of seats and type of car are added. Other difference from Stanford dataset is addition of close up headlight/taillight pictures as well as interior pictures.

Figure 4.2: **Example images from CompCars dataset** Source:[25]

BoxCars116K Dataset is a dataset from 2019 and collected images are from Czech Republic. This dataset is a collection of previous BoxCars21K dataset and new images from traffic surveillance cameras. This dataset contains 116 286 images annotated with manufacturer, model, submodel and year of production.

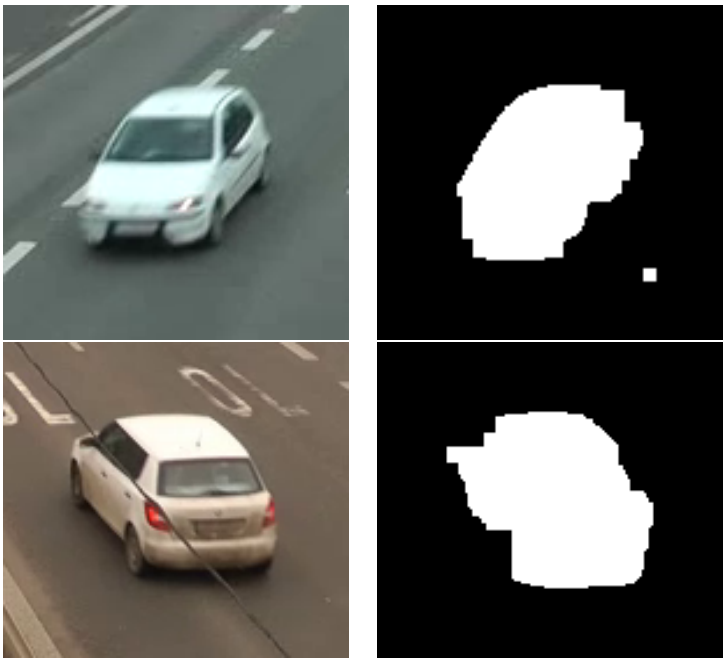

Figure 4.3: **Example of images from BoxCars 116k Dataset, left column shows captured image and right column shows masks of captured vehicles**. Source: [22]

4.2 OpenAI CLIP

CLIP (Contrastive Language-Image Pre-training) is a type of deep learning model developed by OpenAI, first introduced by Radford *et al.* [16] It shows impressive results on a variety of image recognition and retrieval tasks. CLIP model is trained on a large dataset consisting of 400 million image-text pairs gathered from the internet and is publicly available on GitHub from 2021. To guarantee the inclusion of wide variety of visual concepts, the developers used 500 000 queries and up to 20 000 pairs per query to search for image-text pairings. The dataset created through this process is known as WIT (WebImage-Text). It is proprietary and was not made public along with the models.

CLIP uses an objective function designed to encourage the model to map images and text within shared latent space. This makes CLIP particularly good at zero shot learning, which means it can recognize and classify objects or concepts it has not encountered during training. This capability is accomplished by the models shared latent space for text and images, allowing it to generalize from text data image data. For instance, if trained on large dataset of images with its related textual descriptions, CLIP can apply its understanding of the text to form an educated guesses about the content of new images. This ability significantly advances the field of image recognition and retrieval, making it viable for a variety of tasks and industries [3].

For example, in a dataset task that is classifying images of dogs vs cats, CLIP model makes a prediction for each image which text description „a photo of a dog“ or „a photo of a cat“ is more likely to be paired with [17].

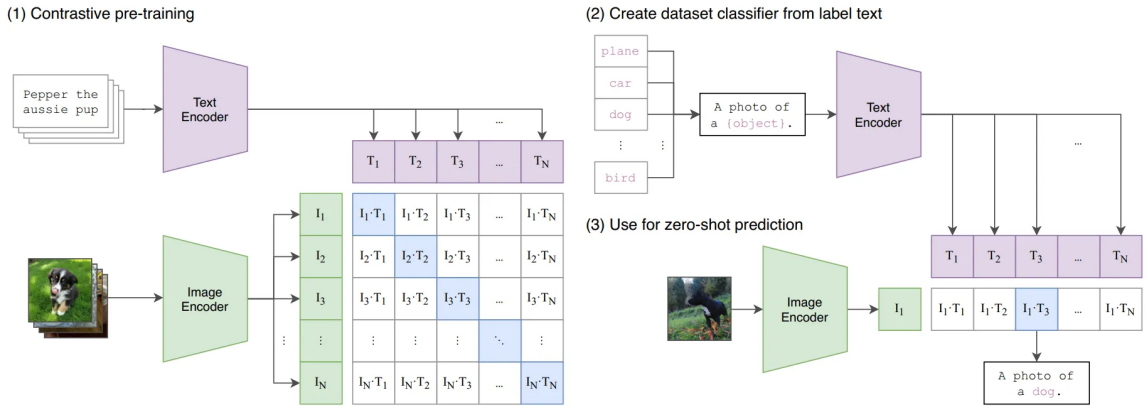

Figure 4.4: **Example of task for CLIP that detects dog in an image.** Source: [17]

Figure 4.4 shows and example of how CLIP works, where part (1) of the image shows how CLIP pre-trains both image encoder and text encoder to predict which texts were paired with which images in the dataset. This training allows to use CLIP as zero-shot classifier. Parts (2) and (3) illustrate how all classes from a dataset are transformed into captions like „a photo of a dog“ and part and how CLIP then predicts which class caption most accurately matches with a particular image [17].

Chapter 5

Implementation

Implementation can be divided into two distinct parts. Firstly, real-time camera worker created to detect and track vehicles using state-of-the-art models. It gathers initial data on detected vehicles and sends them to the rest of the system for further analysis. There can be multiple camera workers running in a system. Secondly, task manager designed to receive detections and distribute them for analysis to other workers that specialize in detecting road conditions, vehicle speed, and vehicle classification. Finalized data are then stored and are accessed with Grafana to visualize their meaning. MQTT is crucial for connecting each part of the system.

This chapter describes the final implementation of traffic surveillance system, detailing each stage of the implementation process. Starting with implementation of real-time vehicle detection, continuing with implementation of supporting programs for additional information and a task manager that keeps it all together. Finally, talks about data storage and capabilities of Grafana for data analysis.

5.1 Technologies

For this task, Python has been chosen as programming language. This was based on prior experience as well as its simplicity while still being able to fulfill everything needed. State-of-the-art YOLO-v8 model by Ultralytics has been chosen for the vehicle detection. OpenAI CLIP was chosen for its ability to extract information from an image. Communication between the system components is provided by MQTT broker. Two different technologies were used for storage, Redis and PostgreSQL.

In order to use these technologies, multiple python libraries were used. Namely Ultralytics for YOLO-v8, supervision for ByteTrack, clip for use of OpenAI CLIP, paho for MQTT, redis and sqlalchemy libraries respectively for use of Redis and PostgreSQL databases.

5.2 Vehicle Detection and Tracking Implementation

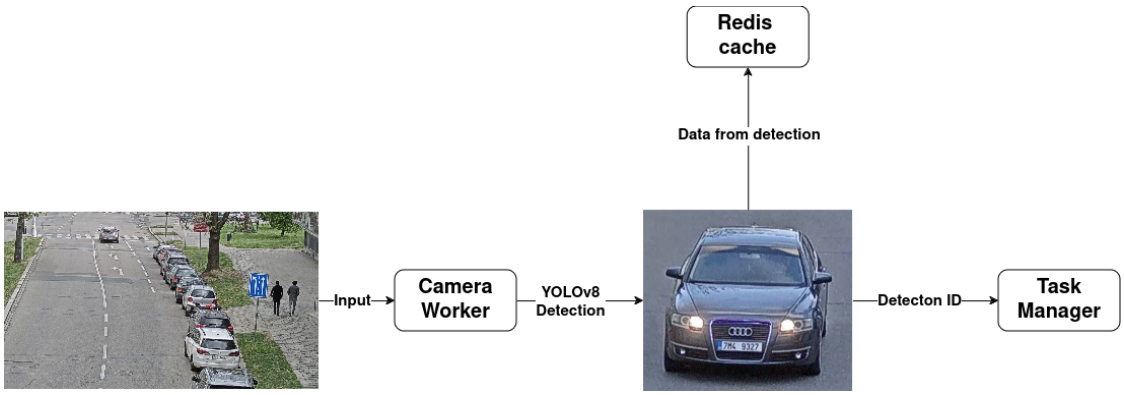

Figure 5.1: **Camera Overview**

Vehicle detection and tracking is implemented as *camera_worker* in Python. The workflow is shown in Figure 5.1. For implementation and testing, AXIS P1435-LE camera situated at BUT university has been used in its native full hd resolution. Additional information about this camera that were needed for speed analysis were provided by thesis supervisor. Each new camera starts by announcing it is live for other workers in a system that require this information through MQTT topic *new_camera*. As part of arguments, detection area is specified as polygon zone. All viable vehicle detections that enter this zone are tracked. Capture position argument fulfills two tasks. It provides the ideal position in scene where the detected vehicle image is taken, additionally if vehicle direction is demanded, it is decided based on x coordinate. Example of detected vehicles is shown in Figure 5.2.

Figure 5.2: **Example of detected vehicles**

Not every camera is positioned to capture enough information about detected vehicles to reliably calculate speed. This is for example in a situation where the camera is set up to only capture the close up of the vehicle. If speed calculation is required, additional information about the camera are required as part of arguments. OpenCV library is used for real-time video processing. For object detection yolov8x model is used and detections are filtered to only contain cars, trucks, and buses. ByteTrack from supervision library is used for tracking the vehicle inside detection zone. When tracked vehicle leaves detection zone, all information about this detection are transformed to be saved as JSON in Redis database.

This JSON contains **ID** of the detection, name of the **camera**, predicted **class** of detected vehicle, **direction** of the vehicle, **entry time** when the vehicle entered detection zone, **camera parameters** that are used for speed calculation together with **speed check** that is 1 if speed will be calculated and 0 otherwise, dictionary of **trajectory** points and dictionary of corresponding **timestamps**, and lastly the dictionary for **image** that contains byte64 string of an image and bounding box for the detected vehicle.

ID of detected vehicle is sent to *task_manager* through MQTT topic named after *camera_id*.

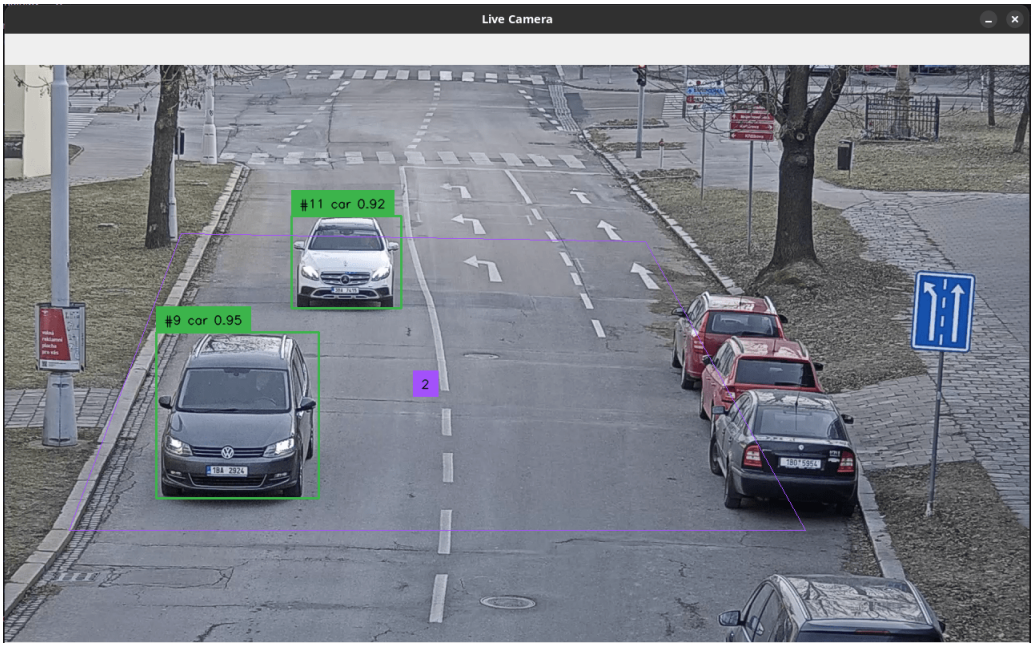

Figure 5.3: **Camera scene showing the detection zone and detected vehicles inside it with tracking ID, predicted class and probability.**

When the *camera_worker* is turned off, it sends MQTT message to topic *camera_done* to notify every worker from the system that kept track of it.

5.3 Data Enrichment and Management

To enrich number of useful information about detected vehicles, additional programs are implemented for various data gathering. In order to keep track of each detection and state of the data enrichment process, *task_manager* is required.

Data transfer and MQTT communication for each detection can be seen in Figure 5.4 where MQTT represented as dotted line and contains ID of the detection is used by *task_manager* to distribute tasks as well as receive confirmation for completed task. Initial detection data are cached in Redis database and each worker requests data needed for its task, new information is then saved back into Redis entry. Task manager takes all data once tasks are completed and saves them to PostgreSQL database.

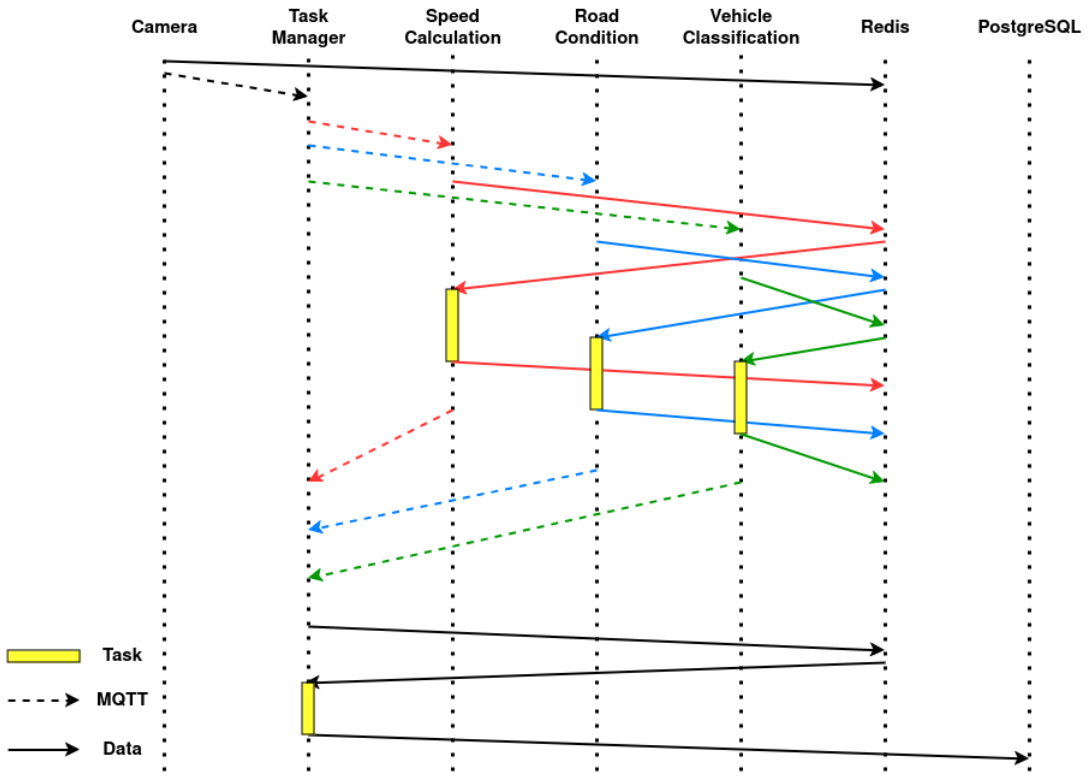

Figure 5.4: **Sequence diagram of each detection passing through the system, red, blue, green representing speed calculation, road condition, vehicle classification tasks respectively. Black represents initial detection communication and final data storage.**

Task Manager

Task manager keeps track of any running camera, new camera is added when it sends its first message to MQTT *new_camera* topic. Task manager then receives detection from cameras through MQTT topics named afted *camera_id*. Every received detection is distributed to the programs for data enrichment using MQTT topics for each task. If the detection does not require speed calculation, this task is not send. Status of these tasks is tracked and their completion is received through MQTT message callback. Once all the tasks are done, cached information about detected vehicle from Redis database is loaded and final data are saved to the permanent PostgreSQL database.

Speed Calculation

Essential functions and classes for camera setup were provided by the supervisor of the thesis. *speed_calc* MQTT topic is subscribed in order to receive tasks. Received task is processed by first acquiring information in form of JSON from Redis about camera that detected the vehicle. These are: camera resolution, camera focal length, camera height and camera horizon position. Camera class is set up using these information. Afterwards, 10 evenly spaced points of trajectory and its timestamps are taken from Redis database and are used to estimate speed of detected vehicle. Trajectory points are taken from bottom center of detection bounding box. Visualization of detected vehicle trajectory is shown in

Figure 5.5. Resulting estimation is saved to Redis detection entry and confirmation of task completion is send to *task_manager*.

Figure 5.5: **Visualization of detected vehicle trajectory used for speed estimation**

Road Condition Detection

Each running camera is saved to keep track of road conditions on multiple cameras. Every ten minutes worker receives MQTT message from camera to the topic named as *camera_id* to update current road condition. Image of the scene is loaded from Redis database and prepared for the recognition. OpenAI CLIP is used for road condition detection, descriptors for these conditions are dry, wet, snowy, and icy. Resulting prediction is saved for that camera. Once a task is received by MQTT from *task_manager*, current road condition on that camera is saved to Redis entry and task completion is sent to *task_manager*. Scene image example used for detection is shown in Figure 5.6.

(a) (b)

Figure 5.6: **Example of the two images used for road condition detection where (a) was detected as „dry“ and (b) was detected as „wet“.**

Aditional Vehicle Information

When the task is received through MQTT, image of detected vehicle is loaded from Redis entry. Example of detected vehicle is shown in Figure 5.7 For vehicle brand and model detection, pre-trained ResNet-152 model trained on Stanford car dataset is used [11]. If detection probability is lower then 80%, its classified as unkown. In addition, vehicle colour is determined using OpenAI CLIP model. Resulting information is saved to Redis entry and task completion is sent to *task_manager* using MQTT.

Figure 5.7: **Example of vehicle image used for brand, model, and color detection, that was classified as: BMW, series 3 2012, gray**

5.4 Data Storage and Visualization

For the purpose of data storage, two different databases are used. **Redis** is used as fast NoSQL cache for detected objects that are being analyzed and their information is not final. Data is store as JSON files and detection ID is used as primary key. This database stores additional information about the detected vehicles such as the trajectory, timestamps, and image of the vehicle saved as byte64 string compared to final database.

PostgreSQL is used as database for final data on detected vehicles as it works well with Grafana. Data is stored in one table containing this information:

- **ID** - is the detection ID, functions as a primary key
- **type** - is the type of detected vehicle predicted by YOLOv8
- **camera** - is the name of camera that detected the vehicle
- **direction** - is the direction of vehicle on two way road
- **brand** - is predicted vehicle brand from additional vehicle classification
- **model** - is predicted vehicle model from additional vehicle classification
- **color** - is predicted vehicle color from additional vehicle classification
- **speed** - is estimated vehicle speed from speed calculation
- **weather** - is detected current road conditions from weather recognition

- **entry time** - is timestamp when the vehicle entered the detection zone

For the implementation testing, this database is implemented using Docker.

Visualization of data from PostgreSQL database is implemented using **Grafana**.

Grafana was running in Docker during development and testing. A custom dashboard was created to display various information about gathered data. These tables and graphs are updated every 5 seconds and show information like the last 10 detected vehicles, number of vehicle types detected over 1 week, or avarage speed for each vehicle type.

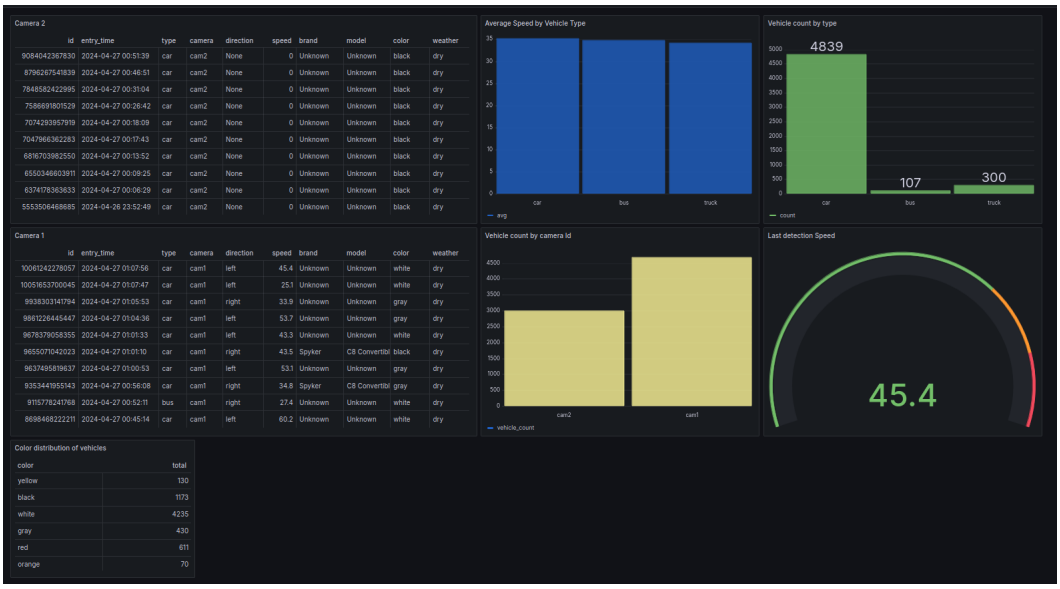

Figure 5.8: **Grafana dashboard**

Grafana dashboard shown in Figure 5.8 creates an interface for all the graphs and tables.

Chapter 6

Results

Data taken from two cameras running on 25.4.2024 can be used to understand traffic that day. *Camera 1* captured 2031 vehicles and *camera 2* captured 1581 vehicles. Dry road conditions were detected for the whole day.

Example of 10 detected vehicles is shown in Table 6.1.

id	entry_time	type	camera	direction	speed	brand	model	color	road
13918412337827	16:26:06	car	cam1	left	31.1	Unknown	Unknown	black	dry
13917089513840	16:26:05	car	cam2	None	0	Unknown	Unknown	white	dry
13915326304276	16:26:03	car	cam1	left	36.7	BMW	3 Series Sedan 2012	green	dry
13914963265951	16:26:03	car	cam2	None	0	Unknown	Unknown	green	dry
13912347945651	16:26:00	car	cam1	left	39.2	Unknown	Unknown	red	dry
13908110533960	16:25:56	car	cam1	left	37	Unknown	Unknown	white	dry
13901816401997	16:25:50	car	cam2	None	0	Unknown	Unknown	white	dry
13899808559714	16:25:48	car	cam1	left	40.5	Unknown	Unknown	black	dry
13896716883136	16:25:44	car	cam1	left	38.4	FIAT	500 Convertible 2012	white	dry
13894561397548	16:25:42	car	cam1	right	23.4	Unknown	Unknown	white	dry

Table 6.1: **Example of detected vehicle Data on 25th April 2024**

As shown in Figure 6.1, the two running cameras detected 3345 cars, 70 buses and 197 trucks.

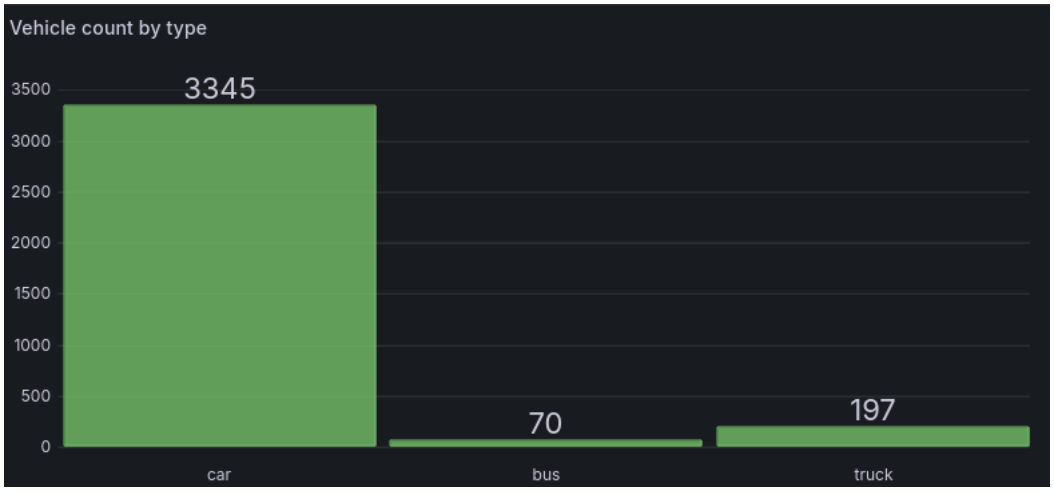

Figure 6.1: **Number of detected vehicle types on 25.4.2024**

Figure 6.2: **Bar graph showing number of detected vehicles on 25.4.2024 during the day.**

Based on Figure 6.2, the highest number of detected vehicles is 497 between 17:00 and 18:00 which corresponds to the time when most of the people are on their way home from work.

Speed was calculated on detected vehicles from *Camera 1*. The total average speed of this day was 33.3 km/h. Highest calculated speed was 70.3 km/h at 12:46 by green Dodge Sprinter Van. Relatively slow average speed compared to the highest measured can be attributed to the position of the camera that is in close distance to an intersection.

Data from *Camera 1* also include the direction of vehicles. on 25.4.2024 there were 1339 vehicles in incoming lane and 692 in outgoing lane. This information can help understand traffic flow.

Most of the detected vehicles are unfortunately classified as unknown. This is because the model used for brand and model classification was trained on Stanford Car dataset that contains vehicles form USA traffic. Number of unknown detections from 25.4.2024 was 2838. The most detected brands were BMW (256), Mercedes Benz (82) and Audi (71).

Statistics about color of the cars can be for example used for commercial purposes. Vehicle colors from 25.4.2024 are shown in Table 6.2.

Vehicle Color	**Count**
White	2067
Black	516
Blue	333
Red	305
Green	167
Gray	150
Yellow	48
Orange	26

Table 6.2: **Color Distribution of Vehicles from 25.4.2024**

For further statistics such as traffic density based on weather, or traffic patterns comparison between workdays, weekends and holidays a longer surveillance is needed.

Chapter 7

Conclusion

The aim of this thesis was to learn about methods used for traffic surveillance and implement a system capable of real-time traffic surveillance from multiple cameras gathering useful information that can be used to create statistics about traffic.

Firstly, it was important to understand what are the benefits of traffic surveillance and what types of useful information can be obtained with various methods.

Secondly, is was necessary to study methods of object detection and tracking, in particular deep neural networks were explored, in order to select the best option for real-time vehicle detection and tracking implementation. Other technologies like OpenAI CLIP and several methods for object classification were studied for additional information gathering.

YOLOv8 model was chosen for vehicle detection and was used in implementation together with ByteTrack designed for tracking of vehicles. Several other scripts were implemented to obtain additional information about detected vehicles. For this, technologies like OpenAI CLIP and Convolutional Neural Networks were used. Task manager was implemented to keep track of the detected vehicles and completed tasks with the use of MQTT broker that provide communication within the system. Redis database was chosen for cache storage and PostgreSQL for final data storage. To visualize and analyze gathered data, Grafana was used.

Results of short system running show it is capable of gathering data from multiple cameras and it is able to communicate with different parts of the system. With the use of Grafana dashboard, data can be efficiently analyzed and studied.

For better understanding of the traffic, a long term system running would be needed. Other possible improvements of the system include training custom model for vehicle classification on dataset made of local traffic and obtaining additional traffic related information such as license plate numbers. More robust system could have possibilities to match vehicles across cameras and find their path through a city.

Bibliography

[1] Amit, Y., Felzenszwalb, P. and Girshick, R. Object Detection. In: Ikeuchi, K., ed. *Computer Vision: A Reference Guide*. Cham: Springer International Publishing, 2021, p. 875–883. DOI: 10.1007/978-3-030-63416-2_660. ISBN 978-3-030-63416-2. Available at: https://doi.org/10.1007/978-3-030-63416-2_660.

[2] Augmented A.I.. *How to Implement Object Detection Using Deep Learning: A Step-by-Step Guide*. 2023 [cit. 2024-04-26]. Available at: https://www.augmentedstartups.com/blog/how-to-implement-object-detection-using-deep-learning-a-step-by-step-guide.

[3] Bhat, A. and Jain, S. *Face Recognition in the age of CLIP & Billion image datasets*. 2023. Available at: https://doi.org/10.48550/arXiv.2301.07315.

[4] Chen, Z., Pears, N., Freeman, M. and Austin, J. Road vehicle classification using Support Vector Machines. In: *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*. 2009, vol. 4, p. 214–218. DOI: 10.1109/ICICISYS.2009.5357707.

[5] Gregor, A. *Rozpoznání výrobce a modelu vozidel [online]*. 2023. [cit. 2024-05-12]. Diplomová práce. Vysoké učení technické v BrněBrno. Available at: https://theses.cz/id/nqiebh/.

[6] Huang, R., Pedoeem, J. and Chen, C. YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers. In: *2018 IEEE International Conference on Big Data (Big Data)*. 2018, p. 2503–2510. DOI: 10.1109/BigData.2018.8621865. Available at: https://ieeexplore.ieee.org/abstract/document/8621865.

[7] Hussain, M. YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. *Machines*. 2023, vol. 11, no. 7. DOI: 10.3390/machines11070677. ISSN 2075-1702. Available at: https://www.mdpi.com/2075-1702/11/7/677.

[8] Icoms Detections s.a. *TMB-134*. [cit. 2024-05-09]. Available at: https://www.icomsdetections.com/project/tmb-134-en/.

[9] Koylu, C., Zhao, C. and Shao, W. Deep Neural Networks and Kernel Density Estimation for Detecting Human Activity Patterns from Geo-Tagged Images: A Case Study of Birdwatching on Flickr. *ISPRS International Journal of Geo-Information*. january 2019, vol. 8, p. 45. DOI: 10.3390/ijgi8010045. Available at: https://www.researchgate.net/publication/

330484322_Deep_Neural_Networks_and_Kernel_Density_Estimation_for_Detecting_Human_Activity_P
Tagged_Images_A_Case_Study_of_Birdwatching_on_Flickr.

[10] Le, B. *Introduction to Multiple Object Tracking and Recent Developments*. 2023 [cit. 2024-05-13]. Available at: https://www.datature.io/blog/introduction-to-multiple-object-tracking-and-recent-developments.

[11] Liu, Y. *Car-Recognition*. 2018 [cit. 2024-05-14]. Available at: https://github.com/foamliu/Car-Recognition?tab=readme-ov-file.

[12] Maity, M., Banerjee, S. and Sinha Chaudhuri, S. Faster R-CNN and YOLO based Vehicle detection: A Survey. In: *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*. 2021, p. 1442–1447. DOI: 10.1109/ICCMC51019.2021.9418274.

[13] Odili, M. *Stanford-cars Image Classification*. 2023 [cit. 2024-05-09]. Available at: https://jovian.com/mitchell-odili/stanford-cars.

[14] O'Shea, K. and Nash, R. *An Introduction to Convolutional Neural Networks*. 2015. Available at: https://doi.org/10.48550/arXiv.1511.08458.

[15] Phung and Rhee. A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets. *Applied Sciences*. october 2019, vol. 9, p. 4500. DOI: 10.3390/app9214500.

[16] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G. et al. Learning Transferable Visual Models From Natural Language Supervision. In: Meila, M. and Zhang, T., ed. *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 18–24 Jul 2021, vol. 139, p. 8748–8763. Proceedings of Machine Learning Research. Available at: https://proceedings.mlr.press/v139/radford21a.html.

[17] Radford, A., Sutskever, I., Wook Kim, J., Krueger, G. and Agarwal, S. *CLIP: Connecting text and images*. January 2021 [cit. 2024-05-11]. Available at: https://openai.com/index/clip/.

[18] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[19] Reis, D., Kupec, J., Hong, J. and Daoudi, A. *Real-Time Flying Object Detection with YOLOv8*. 2023. Available at: https://arxiv.org/abs/2305.09972.

[20] Secure Lane LLC. *Vehicle Detection Loops*. 2013 [cit. 2024-05-07]. Available at: https://www.secure-lane.com/Vehicle-Detection-Loops.html.

[21] Shokravi, H., Shokravi, H., Bakhary, N., Heidarrezaei, M., Rahimian Koloor, S. S. et al. A Review on Vehicle Classification and Potential Use of Smart Vehicle-Assisted Techniques. *Sensors*. 2020, vol. 20, no. 11. DOI: 10.3390/s20113274. ISSN 1424-8220. Available at: https://www.mdpi.com/1424-8220/20/11/3274.

[22] Sochor, J., Špaňhel, J. and Herout, A. BoxCars: Improving Fine-Grained Recognition of Vehicles Using 3-D Bounding Boxes in Traffic Surveillance. *IEEE Transactions on Intelligent Transportation Systems*. 2019, vol. 20, no. 1, p. 97–108. DOI: 10.1109/TITS.2018.2799228. Available at: https://ieeexplore.ieee.org/abstract/document/8307405.

[23] Velodyne Lidar, Inc.. *Velodyne Lidar Powers Intelligent Traffic Management*. January 2021 [cit. 2024-05-10]. Available at: https://www.businesswire.com/news/home/20210111005118/en/Velodyne-Lidar-Powers-Intelligent-Traffic-Management.

[24] Wu, J., Cui, Z., Sheng, V., Zhao, P., Su, D. et al. A Comparative Study of SIFT and its Variants. *Measurement Science Review*. june 2013, vol. 13. DOI: 10.2478/msr-2013-0021.

[25] Yang, L., Luo, P., Loy, C. C. and Tang, X. A large-scale car dataset for fine-grained categorization and verification. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, p. 3973–3981. Available at: https://api.semanticscholar.org/CorpusID:2510693.

[26] Yilmaz, A., Javed, O. and Shah, M. Object tracking: A survey. New York, NY, USA: Association for Computing Machinery. dec 2006, vol. 38, no. 4, p. 45. DOI: 10.1145/1177352.1177355. ISSN 0360-0300. Available at: https://doi.org/10.1145/1177352.1177355.

Appendix A

Contents of the Included Storage Media

- `src/` - Folder with source files
- `latex/` - Folder with LaTeXsource files
- `README.md` - README file for this project
- `environment.yml` - Python libraries dependencies
- `thesis.pdf` - Thesis report file
- `thesis-print.pdf` - Thesis report file for print