Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Rein Nisja Holthe-Berg
Christoffer Skar Lofsberg

Using Computer Vision to Extract Information About the Traffic Using a Vehicle-Mounted Monocular Camera

Master's thesis in Master of Science in Informatics
Supervisor: Jingyue Li
June 2019

NTNU
Norwegian University of
Science and Technology

Rein Nisja Holthe-Berg
Christoffer Skar Lofsberg

Using Computer Vision to Extract Information About the Traffic Using a Vehicle-Mounted Monocular Camera

Master's thesis in Master of Science in Informatics
Supervisor: Jingyue Li
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

This master thesis is dedicated to study hall Alke for making this a fun and sometimes unproductive year.

Abstract

The development and capability of autonomous vehicles have skyrocketed in the last decade. Unfortunately the dream of having traffic consisting of exclusively autonomous vehicles are still not realistic, at least for a couple of more years. In the meantime the need to optimize traffic flow, especially at intersections, are increasing. New vehicles, and especially the autonomous ones, are packed with new and expensive sensor technology. These sensors have the ability of collecting huge amounts of data about the traffic around them, and that data could be used for additional purposes. Like for example managing intersections more efficiently. In this period between now and a future where humans are taken out of the driver seat, it should be possible to use these sensors stuffed vehicles to close the information gap created by vehicles not equipped with these sensors.

Our thesis attempts to use the camera sensors of autonomous and connected vehicles to detect and extract information about other vehicles, and communicate this information to a V2I system on behalf of the other vehicles for intelligent traffic management. The research questions investigate the possibility to use a monocular camera to detect and count vehicles in different lanes, estimate their distance, and compute their speed.

The thesis give evidence that such system has potential, and with further refinements it would be able to confidently provide some of the necessary information to manage intersections more optimally.

Keywords

Computer Vision, YOLO, ITS, Autonomous Vehicles, Connected Vehicles, Monocular Camera, Speed Estimation, Distance Estimation

Sammendrag

Utviklingen og evnen til selvkjørende kjøretøy har økt drastisk det siste tiåret. Dessverre er drømmen om et trafikkbilde som utelukkende består av selvkjørende kjøretøy, fortsatt ikke realistisk, i hvert fall i noen år til. I mellomtiden øker behovet for å optimalisere trafikkflyten, spesielt ved veikryss. Nye biler, og spesielt de selvkjørende, er fullpakket med ny og dyr sensorteknologi. Disse sensorene har muligheten til å samle store mengder data om trafikken rundt seg og informasjonen disse samler inn kan brukes til flere formål. Som for eksempel å styre optimalisere trafikkflyten i veikryss. I denne perioden mellom nå og en fremtid hvor mennesker blir tatt ut av førersetet, bør det være mulig å bruke disse sensorene på selvkjørende kjøretøy for å lukke informasjonsgapet forårsaket av kjøretøyer som ikke er utstyrt med disse sensorene.

Vår masteroppgave forsøker å bruke kamerasensorene til autonome og tilkoblede kjøretøy for å oppdage og hente ut opplysninger om andre kjøretøy, og videre formidle denne informasjonen til et V2I-system på vegne av de andre kjøretøyene for intelligent trafikkstyring. Forskningsspørsmålene undersøker muligheten til å bruke et monokulært kamera for å oppdage og telle kjøretøy i forskjellige kjørebaner, estimere avstanden og beregne hastigheten.

Masteroppgaven gir bevis for at et slikt system har potensial, og med ytterligere forbedringer vil det være mulig å formidle noen av de nødvendige opplysninger for å styre veikryssene mer optimalt.

Preface

This thesis was written as an ending to our master degree in Informatics at The Norwegian University of Science and Technology. We would like to thank our supervisor Jingyue Li (Bill) and Elnaz Namazi for assisting us through the year of writing. We would also like to thank Rudolf Mester and Frank Lindseth for valuable help during the research. Finally we would like to thank friends and family for support through the year.

Table of Contents

List of Tables

List of Figures

Abbreviations

AI Artificial Intelligence 13

ANN Artificial Neural Network 13–15, 23, 25, 26, 51, 79

AV Autonomous Vehicle 6, 7, 9

CNN Convolutional Neural Network 15, 16, 18, 19, 31

CV Computer Vision 13, 16, 18, 21, 30, 51, 91

FPS Frames Per Second 18, 21, 47, 55, 57, 86

HT Hough Transform 25, 26, 38, 40, 78, 79

ITS Intelligent Transportation System 1, 5, 7, 33, 34

LIDAR LIght Detection And Ranging 30, 32, 34, 81

ROI Region Of Interest 25, 32, 39, 40, 79

SAE Society Of Automotive Engineers 7

SVM Support Vector Machine 18, 22

V2I Vehicle-to-Infrastructure 7, 9, 11, 85, 91

V2V Vehicle-to-Vehicle 7

V2X Vehicle-to-X 7, 10

YOLO You Only Look Once 18–20, 38, 47, 51, 57, 67, 77, 78

Chapter 1

Introduction

1.1 Background

Autonomous vehicles have been under rapid development in the last decade and are expected to become a significant factor in the traffic in a couple of years [1]. At the same time, non-autonomous vehicles have become more and more connected to the infrastructure.

The use of different sensors and automated systems, such as cameras and driving assistants, has become more frequent the recent years. These sensors could potentially be utilized to provide necessary information to an Intelligent Transportation System (ITS) which aims to create better traffic flow, reduce congestion, and decrease pollution in cities.

1.2 Motivation

The motivation behind this thesis was to utilize sensors mounted in a vehicle, more specifically a camera to extract information about the traffic situation in the vicinity of a vehicle. This information could then be communicated to an ITS to increase its performance. To be able to do this, a system that uses this data and analyzes it with state-of-the-art image processing techniques had to be built.

1.3 Research Questions

To build such aforementioned system, we designed three research questions to help us solve parts of the bigger solution. These questions were as follows:

RQ1: How can the number and type of vehicles in nearby lanes of a vehicle be found using image data captured from a vehicle-mounted monocular camera?

RQ2: What is the most accurate combination of width and height when calculating distance using image data captured from a vehicle-mounted monocular camera?

RQ3: How can speed of the nearby vehicles be estimated by using distance calculated from image data captured by a vehicle-mounted monocular camera?

1.4 Research Method

The research method used was the Design Science Research Process, designed by Peffers et al. [2]. This method was a six-step method that was built on the design and development of a system to solve the overall research goal.

1.5 Research Results

The general result is a working proof of concept, which utilized state-of-the-art object detection algorithms and computer vision methods, to provide information to a V2I system for intelligent intersection management.

To evaluate the performance of the system with respect to RQ1, the output of three different traffic scenarios were compared with the manually counted numbers for each scenario. The vehicle detection demonstrated promising numbers, with a detection error of only 7.6%. This number does not take vehicle type into consideration and was merely a measurement of the ability to detect objects. When also strictly evaluating vehicle type, the error increased to 38%, mostly due to misclassification of vehicle type and wrongly detected lane lines. Overall the result showed that the system was not performing as good as hoped, but with some further work it could achieve satisfiable results. The object detection algorithm could perform better with more fine tuning and more specific training of the ANN, and a more advanced lane detection approach could improve the lane detection.

Finding the most accurate combination of height and width for RQ2 was done by measuring stationary vehicles. The vehicles were recorded at different distances and measured by a laser to know the true distance. The system computed the distance using different ratios of height and width, and the output was compared with the ground truth from the laser. This experiment showed that the most accurate ratio between height and width was a distribution of 85% of the height and 15% of the width. Limitations of the experiment was that the vehicles were stationary, and future work would be to validate the findings with ground truth on real traffic measurements.

To estimate speed of other vehicles for RQ3, the system used distance calculated with the ratio found in RQ2. To evaluate the performance of the system, the ”true” speed of the target vehicle were manually calculated and compared with the system output. This experiment gave evidence that computing speed of vehicles in front of the camera was feasible when using estimated distance from object detection and a monocular camera. Improvements of the other aspects of the system would yield more stable measurements and could improve the speed estimation further.

1.6 Thesis Structure

The thesis is structured in the following way:

- **Chapter 1 - Introduction:** This chapter presents the outline of the thesis.
- **Chapter 2 - Background:** This chapter presents the background theory and required information for the thesis.
- **Chapter 3 - Related Work:** This chapter presents the state-of-the-art technology and work already done in the field.
- **Chapter 4 - Research Design and Motivation:** This chapter presents more in-depth the motivation, research goals, research questions, and the research method.

- **Chapter 5 - Research Implementation:** This chapter presents the research implementation used to answer the research questions. Here technology choices for the system are described.
- **Chapter 6 - Research Results and Evaluation:** This chapter presents the different results generated by the implementation of each research question. The results were also evaluated here.
- **Chapter 7 - Discussion:** This chapter presents the results and evaluation.
- **Chapter 8 - Conclusion and Future Work:** This chapter presents the conclusion of the thesis and future work.

Chapter 2

Background

This chapter present the background information needed to understand the overall problem and motivation for this thesis. The chapter gives an introduction to intelligent intersection management, and autonomous vehicles.

2.1 Intelligent Traffic Management

An ITS is by the European Union (EU) defined as

> Advanced applications which without embodying intelligence as such aim to provide innovative services relating to different modes of transport and traffic management and enable various users to be better informed and make safer, more coordinated and ‘smarter’ use of transport networks[3].

These systems are used widely in the world today, and in intersection management, they can vary from just using sensors [4] to more complex systems where machine learning and image processing are in use [5].

2.1.1 Signalized Intersections

Light signals have been the primary method for managing traffic flow in intersections for many years. The traffic regulated intersections are mostly using timers to control the flow with a fixed green light interval. If pedestrians are taken into account, there is often a button that interrupts the fixed interval and gives the green light to the pedestrians.

There are different problems with how the most common signalized intersection handles the traffic. Engineers, researchers, and governments are continually trying to overcome as much congestion as possible, by for instance, making the light intervals more efficient [6, 7].

Use of Inductive Loops in Intersections

One of the oldest and most used approach to control intersections in high-density areas is to use inductive loops built into the road [4]. This has proven to have good results regarding the counting of vehicles at the intersections with accuracy on around 98% [8]. However, the installation of inductive loops can be expensive as they have to be built into the road, and maintenance costs on a faulty sensor will cost the equivalent amount. A faulty sensor will also often results in the intersection allocation the maximum green light possible, and this can cause serious congestion in the intersection [9]. The loops can also not be replaced in cold weather [10].

Use of Cameras in Intersections

A newer approach to intersection management is to use cameras to count and track the number of vehicles waiting at an intersection. This is an approach that has been around since the early 90s [11], and in the 2000s, the state of Texas developed a manual for installing cameras in intersections [12].

In recent years, when object detection has become more prevalent in computer science, this has also become a lot more common in intersection management. Moshiri et al. made a comparison with different intersection management systems [10], and discovered it could be hard for the camera systems to be accurate with the change of light and weather conditions. The systems were useful in clear weather, but the accuracy declined with worse weather and especially when it was snowing.

2.2 Autonomous and Connected Vehicles

2.2.1 Autonomous Vehicles

Research on Autonomous Vehicles (AVs) has been conducted since at least the 1920s. In the early stages of development, AVs had some form of guidance through electric or magnetic cables embedded in the road [13]. Developing fully autonomous vehicles has proven to be a challenge. Compared to other types of vehicles, such as planes and boats, there are more variable factors to take into consideration when driving a car. Other drivers and vehicles, pedestrians, and varying road conditions are some of the challenges faced by AVs [14, 15].

With the development of many different automatic features for driving, it was necessary to classify to which degree the system is involved in the driving task. Society of Automotive Engineers (SAE) has developed a standard for levels of automation in AVs in SAE (J3016) [16]. The standard has six levels, ranging from 0 to 5, where 0 is no automation at all, and 5 is full automation. Table 2.1 shows how the different levels of automation work and what they demand of an autonomous system.

Most of the new vehicles today have systems on board that places them in automation level 2 or 3. Sensors, such as LIDAR, RADAR, Ultrasonic sensors, and cameras, are installed in many of the most advanced car models. Almost every vehicle manufacturer are working towards delivering a car that satisfies the level 4 or 5 requirements. It is a race to be the first vehicles manufacturer that delivers this, and many of them claim in 2018 they will have a commercial product within the next 3 to 5 years. This means that the car industry is working hard on research and development. Everyone thinks that fully autonomous cars are the future, and everyone wants to be a part of the journey.

Big manufacturers such as Volvo [17, 18], BMW [19], Ford [20, 21] and Tesla [22, 23] are currently installing equipment in their vehicles to increase the level of automation and they are approaching their goal quickly. Tesla has a working autopilot today, where the vehicle can drive by itself in certain scenarios just based on cameras, sensors, and other hardware installed in the vehicles [22, 23, 24].

2.2.2 Connected Vehicles

A connected vehicle is a vehicle that is wirelessly connected to internal and external environments [25]. The connected vehicles can be connected in different environments, such as Vehicle-to-Infrastructure (V2I), Vehicle-to-Vehicle (V2V) and Vehicle-to-X (V2X) to create better ITSs and increase the level of automation of the vehicles [26].

Automation Level	Name	Description
Level 0	No Automation	Human driver is responsible of all aspects of the driving task
Level 1	Driver Assistance	Human driver performing all tasks, but having a system assisting with either steering or acceleration/deceleration
Level 2	Partial Automation	A driver assist system performs the task of both steering and acceleration/deceleration by using information about the environment. The human driver must be monitoring the driving environment and be ready to take over
Level 3	Conditional Automation	Automated driving system perform all aspects of the driving task, but the human driver must be ready to respond and intervene when needed
Level 4	High Automation	Automatic driving system perform all aspects of the driving task. Can handle most situations even if the driver does not respond to requests to intervene
Level 5	Full Automation	Automatic driving system perform all aspects of the driving task. Human intervention is not necessary

Table 2.1: Levels of Automation according to SAE [16]

2.3 Intersection Management using information from Autonomous Vehicles

As mentioned in Section 2.1.1, there has been research and proposed solutions on creating more dynamic flow in intersections, which can increase the throughput of vehicles. There are already different solutions today that utilize the benefits of AVs and use their reliability and non-biased driving to create solutions based on V2I communication [27, 28]. These solutions, however, are possible when all the traffic consist of AVs, and human control is not present, as humans are unpredictable.

An example of a solution that relies on V2I communication and fully autonomous vehicles are the solution presented by Chouhan and Banda in 2018 [27]. All vehicles were assigned a slot by a central vehicle scheduler, and all traffic had to drive in the corresponding lane related to their destination. By knowing this lane restriction and assuming all vehicles listened to the vehicle scheduler, they were able to create a system that controlled an intersection automatically.

2.4 The use of Information from Autonomous and non-Autonomous Vehicles Together

Connected vehicles do not need to be autonomous to be connected. There are several situations where both AVs and manually steered, but connected vehicles, can communicate to create better traffic flow.

Priemer and Friedrich presented in their paper a new and innovative way to improve the traffic flow in signalized intersections by using V2I communication [29]. With Dynamic Programming and Complete Enumeration in an algorithm, they attempted to solve the next 20 seconds of queue length quicker by using earlier seen examples. The proposed solution was to install communication devices in the vehicles to create connected vehicles and then let the vehicles provide information about their ID, position, and velocity data to the system. The algorithm also utilized data from inductive loop sensors placed in front of the stop line, as these provided real-time data of vehicles that were standing in the intersection.

A problem with these solutions today, and this was also something that Priemer and Friedrich experienced, was the number of vehicles that have a communication device installed [29]. They were dependent on a high penetration rate and could only rely on vehicles with these installed. The effect of the system decreased significantly when the penetration rate dropped.

2.4.1 Communication Standards

To be able to use V2X communication to exchange data with connected vehicles, a few standards for communication was needed. The most significant standards today are developed by European Telecommunications Standards Institute (ETSI), Institute of Electrical and Electronics Engineers (IEEE), and Society of Automotive Engineers (SAE). These standards can vary based on the location of the infrastructure and vehicles, but they are quite similar.

In the United States, a spectrum band is dedicated to V2X communication. This is called the Dedicated Short Range Communication (DSRC) spectrum band [30]. Figure 2.1 shows how the DSRC communication bands are divided into different channels allocated for different purposes when communicating V2X.

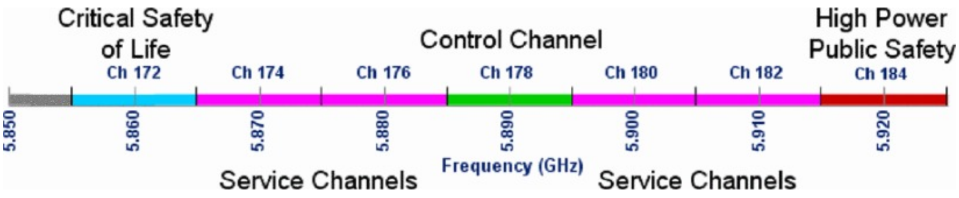


Figure 2.1: The figure shows how the DSRC spectrum is dived into separate channels for different types of communication. Certain channels are regulated, but the use of the DSRC band is free as long as the communication type is related to the specific band's purpose [30].

This network band in the US is regulated by multiple standards. One of them is the IEEE 1609 WAVE, which stands for Wireless Access in Vehicular Environment [31]. This standard works alongside the IEEE 802.11p to create a reliable communication service that is easy to use [31]. The WAVE architecture is complex and brings different IEEE standards together to create a complete communication system. Figure 2.2 shows how the WAVE architecture brings all these standard together to create a fully working V2X system.

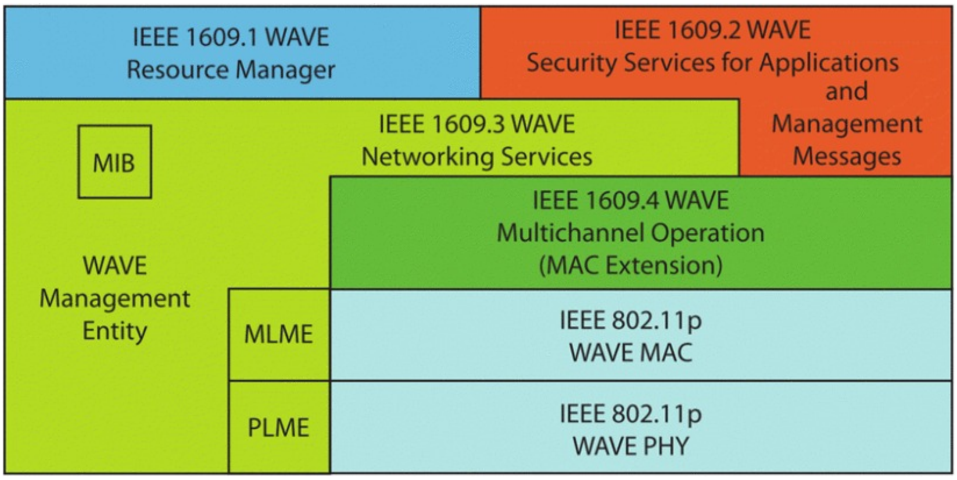


Figure 2.2: This figure shows an overview of WAVE and its different components. This show how IEEE 809.11p can relate to WAVE and be used as a communication standard V2X. The different components are used for different parts of the system, such as resource management and security [31].

There are multiple other standards developed. One of them is the ETSI TS 302 637 [32], which is a similar standard developed for Europe. There is also another type of standards developed, such as eCoMessages, which focuses on an environmental effect [33].

2.5 Required Information for Intersection Management

To create an intersection system which can utilize data from connected vehicles with V2I communication, there is a need for specific information about the traffic that approaches. Table 2.2 display the essential information which is required [27, 34].

Data:	**Description:**
Vehicle ID	An ID of the vehicle in the intersection
Source lane	The entering lane of the vehicle
Destination lane	The destination of the vehicle
Entering velocity	The velocity of the vehicle when entering the intersection
Vehicle length	The length of the vehicle
Vehicle acceleration	The capability of acceleration of the vehicle
Position	The position of the vehicle in longitude and latitude

Table 2.2: Required Information Needed to Create a V2I Intersection Manager

Chapter 3

Related work

This chapter presents the state-of-the-art of Computer Vision. Further, it elaborates how Computer Vision are utilized in traffic and vehicle-related affairs.

3.1 Computer vision

Using images or video from cameras on a vehicle to extract information about the surrounding traffic requires the computer system to see and understand the traffic. Making a computer see and understand the same way humans do is not a simple task. Computer Vision (CV) has been a research topic for many years and is still a difficult problem to solve [35]. A major contributor in recent years development is the availability of computing power, with the increasing power and decreasing cost of graphics processor units (GPUs). This enables the utilization of deeper Artificial Neural Networks (ANNs) to perform the calculations needed to identify objects in images, and solve other Computer Vision tasks. Computer Vision is a subfield of Artificial Intelligence (AI), and this section will elaborate how CV utilize AI and machine learning to make computers see.

3.1.1 Artificial Neural Networks

ANNs is one of the most common ways to do machine learning today and has been around for many years. ANNs has gained traction in the last decade, due to the development and availability of hardware with good enough performance. As Warren Sarle wrote in his paper [36], the idea was to replicate the brain with concepts collected from the human brain. A basic model takes input and processes the value based on different weights that are related to the different ”neurons” in the system. An output is generated based on activation functions in the layers of the network, and the last layer returns the output. Figure 3.1 describes a simple ANN with three layers without any activation functions or weights included.

In order to receive accurate data, the network has to be trained. This is done by adjusting the different weights in the network with a factor generated from a loss function that processes the error from the previous run.

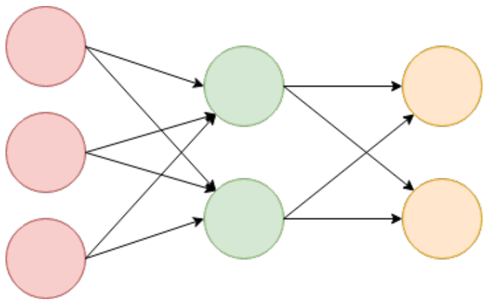


Figure 3.1: Illustration on how a basic artificial neural network is structured. The red layer works as an input layer with three different inputs. These values are processed in the green hidden layer with different weights on the connections between the neurons. Finally, the yellow output-layer produces two different outputs.

3.1.2 Deep Learning

Deep learning is a method of doing machine learning with ANNs where networks with many layers are used. Shallow networks have been around for decades, but the ability to do computations on deeper networks have not been possible until years when enough computational power became more available [37]. It is possible to do deep learning on many types of ANNs, such as feed-forward networks, networks with backpropagation, and Convolutional Neural Networks. The latter works great on images and other two dimensional data.

3.1.3 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a specialization of an ANN. The use of these networks has increased in the last couple of years, mostly due to the ability to successfully detect objects in images. The main difference between normal ANNs and CNNs are the use of convolution instead of general matrix multiplication [38].

Convolution

The definition of a CNN, is an ANN which uses convolution in at least one layer of the network. Convolution is an operation that combines two functions and produces a new function with smoothed results between the two earlier functions. CNNs use convolution with a function often referred to as the kernel, with the weights and the input, which is values from an image. Images are often in multiple dimensions and the mathematical equation 3.1 describes a two-dimensional image with a two-dimensional kernel [38]. In this equation, I is the input image, K is the two dimensional kernel, i and j are coordinates in the image array. The output is the the value of the specific part of the image with the kernel taken into consideration.

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(m,n)K(i-m,j-n) \tag{3.1}$$

Equation 3.1 makes it possible to do convolution on regions of the image and at the same time reduce the number of computations needed to find edges. Where convolutional layers use the convolutional formula, regular layers often use matrix multiplication. Convolutional nodes with different kernels will find different features in the image.

Pooling

Another important feature of a CNN is the pooling layers across the network. These layers will pool results from the network together and merge previous outputs into one value. One of the most used pooling techniques in CNNs is max pooling, which takes the result of the node with the highest output and transfers this result further. Figure 3.2 shows how this is done in practice. When using this on images, each bottom layer node will typically have a feature, and the layer will forward the value from the most likely correct feature [38].

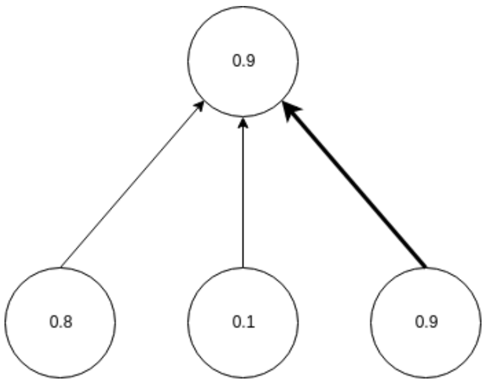


Figure 3.2: Illustration of how a pooling layer works when using max pooling. The node will forward the highest input and discard the others.

3.1.4 The Use of Convolutional Neural Networks on Images

As mentioned, the use of CNNs has increased the last couple of years and contributed to the breakthroughs in object recognition and CV. The use of these networks has been good at recognizing different objects in images due to the use of pooling and different convolutional layers. Figure 3.3 describes how a CNN works on a small picture to recognize the eye of a dog. The different features of the image are extracted based on different similarities, and then different color values are extracted based on the RGB values of the eyes [39].

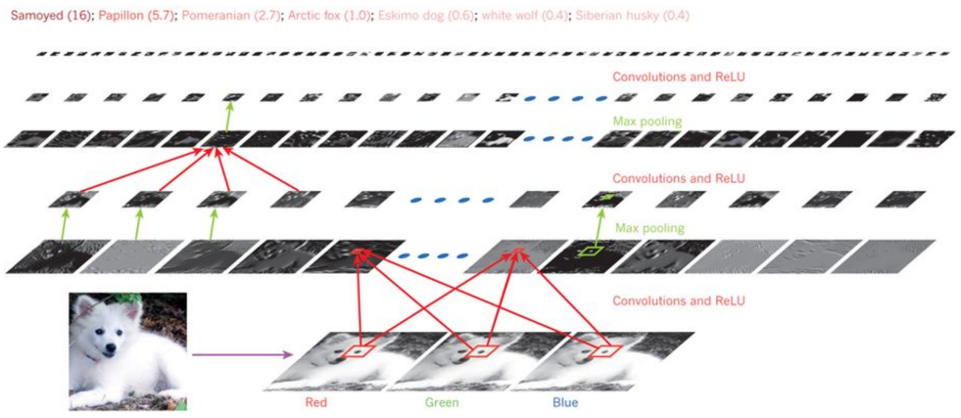

Samoyed (16); Papillon (5.7); Pomeranian (2.7); Arctic fox (1.0); Eskimo dog (0.6); white wolf (0.4); Siberian husky (0.4)
Convolutions and ReLU
Max pooling
Convolutions and ReLU
Max pooling
Convolutions and ReLU
Red
Green
Blue

Figure 3.3: Illustration of a Convolutional Neural Network. First the image is split up into an array of red, green and blue values, and then these are pooled together into features. Each image is a feature map generated by the convolutional layers. [39]

3.2 Detecting Vehicles with Computer Vision

Object detection is a significant part of CV and crucial for autonomous vehicles to be able to operate. They must be able to detect objects related to traffic such as vehicles, pedestrians, signs, traffic lights, but also other objects that may interfere with the process of driving. For use in autonomous driving, the object detection must be of high accuracy, but also high computational speed. Real-time performance is required, which means as fast as the image source provides input. Most video cameras operate at 30 Frames Per Second (FPS), which means a processing time of 33 ms per image. To achieve this, the hardware often requires a GPU and a highly optimized object detection system.

3.2.1 Convolutional Neural Network Implementations

One of the big breakthroughs in object detection came with the introduction of AlexNet in 2012. AlexNet uses CNNs to classify images on the ImageNet dataset and is often considered the father of the big CNNs of today [40]. One of the biggest reasons for AlexNets success was the utilization of multiple GPUs, ReLU, and dropout [41]. After the success of AlexNet, many improvements have been done with object detection and CNNs the last couple of years up until today's state of the art.

R-CNN is one of the algorithms that obtained the status of state-of-the-art right after AlexNet in 2014. The algorithm utilized the properties from AlexNet to do the feature extraction from images, before several Support Vector Machines (SVMs) did the final classification [42]. This was more efficient than AlexNet and other algorithms of the time, as they used multiple datasets for training such as Pascal VOC [43], and not only ImageNet, which was the major dataset of the time.

3.2.2 YOLO - You Only Look Once

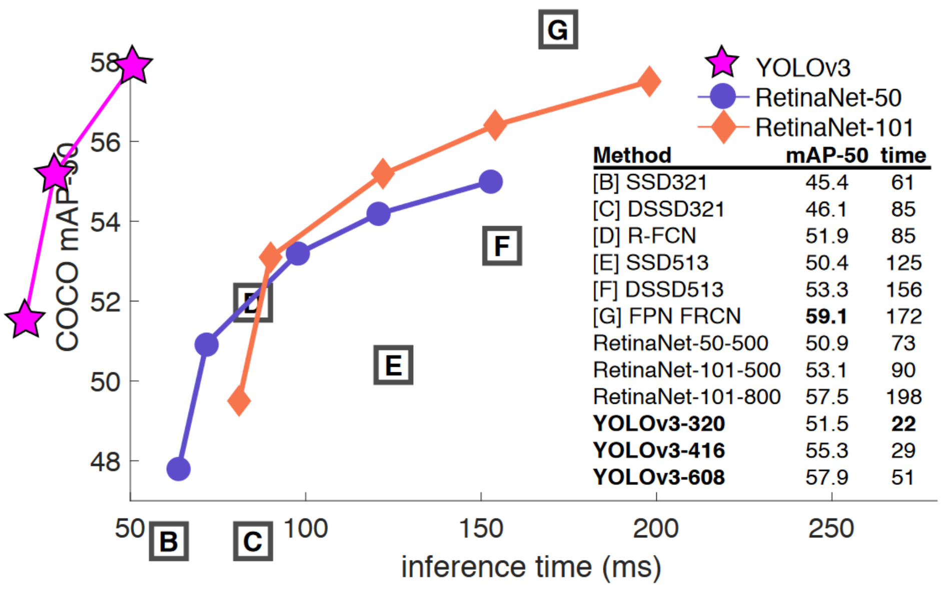
You Only Look Once (YOLO) was a relatively new approach to the task of object detection in CV. The algorithm was published in 2016 and was the state of the art of object detection. One of the main advantages over other CV algorithms was the speed, making it very suitable for real-time usage. Previous methods used classification to perform the detection, but

YOLO used bounding boxes and regression to associate class probabilities to possible objects [44].

Multiple algorithms were based on classification [42, 45]. They perform multiple steps to perform the object detection task, where the first step often is to find parts of the image that may contain an object. When all interesting regions are found, the second step is to try to classify each region individually to find which object is present. Detection systems before YOLO evaluated a classifier for an object at different locations and scales in an image. Running a classifier over an image is recognized as a sliding window approach. Another way to do detection is to first propose potential bounding boxes around possible objects and then run classification on these boxes. Post-processing is done to remove duplicate detections and refine the bounding boxes. R-CNN uses this region proposal method [42]. The entire pipeline is complex and hard to optimize since the individual parts must be trained separately. This also causes speed and performance to suffer.

YOLO differs from these classification approaches by basing the detection on regression. Instead of first finding parts of the image and then do classification, YOLO propose bounding boxes around possible objects and calculates the probability distribution of which classes might be present within that bounding box. This makes the process much faster because the image is processed only once in a run through the algorithm. The class that has the highest probability score is selected as the object in that bounding box. The minimum probability for a class also must be above a certain threshold to be selected as an object and for the bounding box not to be discarded. YOLO uses non-max suppression to overcome the issue with overlapping bounding boxes detecting the same object.

YOLOs way of doing detection proposes several advantages compared to older algorithms. In addition to having significantly higher speed compared to other object detection algorithms, the detection pipeline is less complex. This causes less computational load due to the processing of an image being done only once. It is also fast to do detection and classification when the network is already trained. The detection accuracy is also on par with other state of the art algorithms, such as Faster R-CNN, but YOLO is up to 10 times faster. The drawbacks of YOLO are the slightly lower accuracy and the weakness to smaller objects and small input images. The weakness of smaller objects has been rectified with the latest iteration of the algorithm [46].



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Figure 3.4: YOLOv3 Comparison [46]

One must look at the use case for an algorithm to determine if the drawbacks are worth the advantages. In object detection for use in traffic; time is of the essence, and the slightly lower accuracy of YOLO is worth the speed-up. For use in real time vehicle detection, the YOLO algorithm is therefore very suitable. [44, 47, 46]

Zhou et al. used YOLO in their paper where they looked at ”...vehicle detection and classification problems using Deep Neural Network approaches” [48]. Some of their work revolved around how to utilize Deep Neural Networks for vehicle detection, and they demonstrated that YOLO gave good results. In YOLO version three, the algorithm also scored significantly better than other comparable algorithms such as RetinaNet [49]. Figure 3.4 show how YOLO compares to other algorithms..

3.2.3 Keeping Track of Detected Vehicles

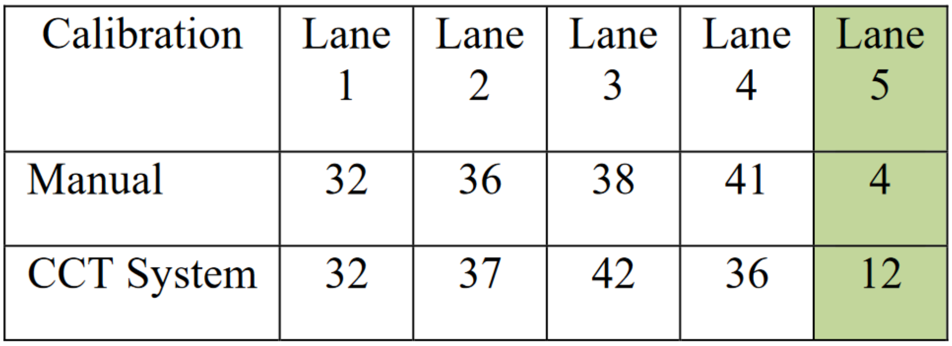
Often it is not enough to only detect a vehicle in an image. If the input is a video, it is desired to track the vehicles between multiple frames, identifying the same vehicle even though it may have moved. By tracking vehicles, it is possible to calculate more information that is not directly detected, such as the vehicles speed.

Most of the research where the goal is to keep track of multiple vehicles with a camera is done with stationary cameras filming the traffic [5, 50, 51]. Chang and Wong managed to keep track of different vehicles in an intersection by using the pedestrian crossing as a size reference to calculate the distance and size, and used the FPS to calculate speed [5]. They used the optical flow method to do the calculations, and the features extracted were used to track the movement of each vehicle and classify them into different categories. Some of the results they achieved were the findings of different driving patterns between non-stopping vehicles with green light, vehicles that stopped at red light, and vehicles that just slowed down. Why a vehicle slowed down was difficult to tell. All this data was used to manage the intersection.

Akoum wrote in 2017 about another approach where the goal was to count vehicles in an intersection with CV using edge detection [50]. The first method he provided was a filter method on a video stream, where he used a filtering technique to detect the foreground and background of the image to detect the vehicles. He used Gaussian Mixture Models, changed the colors, and applied filters to do this. After the foreground was detected, the image was processed to remove noise and all objects that had fewer pixels than a given threshold. The three last steps were about the detection of vehicles, tracking, and counting. To detect the vehicles, he looked at the change in the picture since he dealt with moving vehicles. For tracking, he used a Foreground Detector Blob Analysis function. This detected vehicles and drew rectangles around them. To count the vehicles, he just counted the bounding boxes around the vehicles.

This methods approach was similar but less naive than the method Pancharatnam and Sonnadara proposed in 2008 [51]. Here they looked at determining the performance of different processing techniques. Where Akoum filtered out the background with much processing, Pancharatnam and Sonnadara looked at the video when the pixels were stable, and no vehicles were present, to extract the background. They could then identify vehicles by subtracting the background from frames containing vehicles.

A projective transformation matrix was used to map image coordinates to road coordinates [51]. This calibration was necessary to be able to find correct values for speed. They made the assumptions that the road was straight, flat, that the traffic flow corresponded to the y-axis, and the x-axis was perpendicular to the traffic flow.



Calibration	Lane 1	Lane 2	Lane 3	Lane 4	Lane 5
Manual	32	36	38	41	4
CCT System	32	37	42	36	12

Figure 3.5: Results of Vehicle Counting in Separate Lanes from Pancharatnam and Sonnadara's Research [51]

The goal of their project [51] was to identify vehicles in separate lanes. Lanes were identified by their center line, and vehicles were tracked along this center line for each lane. Vehicle detection and tracking were done by identifying pixel value change along the proposed center lines and using the values in a binary column vector. If a column of 1-values were long enough to be classified as a vehicle, it was tracked over multiple frames.

To test the counting capabilities of the system, they compared the result of the system with a manual count. The camera was positioned on a bridge above the traffic on a highway. Figure 3.5 display their results.

Their research confirmed that a system was able to efficiently and reliably track vehicles. With optimal camera calibration, they were able to get results above 90% accuracy. For lanes furthest to the side, causing more distortion in the image, the results were considerably worse.

The second approach Akoum developed was about detecting the vehicles in a single image from a stationary camera [50]. He used edge detection to find the edges of the vehicles in the image and then used the image created to track the vehicles. The techniques proposed by Akoum could be implemented quite cheap because only one camera per intersection was needed. His experiments showed an accuracy of around 90% when counting the vehicles both on video and by image.

Raj Uppala demonstrated in 2017 how he used a SVM to perform vehicle detection and tracking [52]. He used Histogram of Oriented Gradients for feature extraction on a labeled dataset to train a linear SVM classifier. The sliding window technique was used to identify vehicles in an image. Tracking was done by generating a heat map of detections for each frame in a video sequence and identifying recurring detections over multiple frames. Uppala's conclusion was that the method to identify and track vehicles worked ok, but the drawbacks were the performance. A slow and computationally expensive pipeline made the method not very suited for real-world scenarios. His suggestions for a better solution were to use an ANN or the YOLO algorithm.

3.3 Detecting Lanes

Several ways to detect lanes from images has been under development for years. Detecting straight lane lines is easier than detecting curved, and a greater challenge is detecting lanes when the markings are old and worn out [53]. This section will focus on different techniques of detecting lanes from a vehicle.

3.3.1 Detecting Edges in Images

The first step to detect lanes is to find the edges in the image. These edges are the areas where different parts of the image meet, and the color change is over a set threshold. Multiple algorithms can be used to find these edges, and those presented here are some of the most common used.

The Sobel Edge Detection algorithm utilizes an operator called the Sobel operator, which is an estimation to a derivative of the image values [54]. By using a 3x3 matrix for both x and y coordinates, the operator is applied through the whole image and differences are calculated, with a focus on the pixels that are connected to the origin of the operator. If the difference is above a threshold, an edge is found.

The Prewitt Edge Detection algorithm [55] is similar to the Sobel Edge Detection algorithm. This algorithm also has an operator that is used in the process, called the Prewitt operator. This algorithm uses the vertical and horizontal lines in the image by looking at differences in pixel intensity. A 3x3 matrix is also used here, but there is no focus on the pixel that is touching the origin of the matrix.

The Canny Edge Detection algorithm [56] has a multiple step approach. First, it first uses Gaussian filters to smooth the image, This is to remove noise that can act as edges, but it is essential not to remove too much. The algorithm then calculates the gradients of the image using an operator which for instance could be the Sobel or Prewitt, operator. Finally, the algorithm suppresses non-maximum values and then use a threshold to find the values that represent the edges in the image.

These algorithms are all reliable algorithms that are commonly used. There is done multiple comparisons of different algorithms, and they are all good on different scenarios [57] [58].

3.3.2 Detection of Lanes

A typical edge detector when dealing with lanes is the Canny Edge Detector [59, 60, 61] and then use a variation of Hough Transform (HT) to detect lines from the edges [60, 62]. This subsection describes different approaches used to detect lanes.

The first approach was to use HT to detect the lines. As mentioned, this was a common method that is widely used. Satzoda et al. [62] and Deng and Wu [60] used different variations of HT to detect straight lines in an image. HT consists of two steps, ”voting” and ”peak decision.” In the voting step, each edge pixel $P(x, y)$ is transformed into a sinusoidal curve with equation 3.2. In the equation, p is the length of the perpendicular line passing through (x, y), and θ is the angle made by the perpendicular line with the x-axis.

$$p = x \cos\theta + y \sin\theta \tag{3.2}$$

HT is quite resource consuming as it has to be done on every pixel in the edge map. Satzoda et al. proposed another solution which they called Hierarchical Additive Hough Transform, where they grouped pixels instead to reduce the resource cost. Another variation of HT is the Progressive Probabilistic Hough Transform [63]. This variant instead chooses random points in the edge maps for voting and analyze if the probability of the point being noise or not. A threshold is set to ensure that false positives and negatives are not discovered. This was a more inaccurate approach than the standard HT, but it reduced the computational time.

The use of Region of Interests (ROIs) was also an approach that was used to minimize the computational power needed to detect the lanes [60]. This was an approach that removed unnecessary areas in the image, to reduce computations needed, as the lanes could be assumed to always be on the lower part of the image.

Kim developed a method that could detect lanes in several steps which followed the ”hypothesize and verify” paradigm [53]. The method consisted of rectifying the image, detecting possible lanes in the image before grouping this into lane hypotheses of where the system thought where the lanes were. These hypotheses were taken as input into an ANN that selected a right and a left lane. To detect the lanes in the rectified image, the image was first converted into grayscale to make the computation easier, and then the algorithm detected the possible lane pixels which were grouped into uniform cubic-spline curves. The fitting algorithm that was detecting lanes grouped these curves into line segments and used Gaussian smoothing to remove noise. Everything was scored by an ANN, and the highest score was chosen to be the left and right lane.

Methods using HT has received great results on straight lines where Deng [60] and Wu [64] managed to detect 98% of the proposed lines whereas Kim [53] managed to detect around 80% of the lanes fed into his algorithm.

3.4 Distance Estimation

3.4.1 Estimating Distances in Images

Calculating distance to an object in an image is easy if all information about the equipment and the actual size of the object is known. To find the distance to an object, the Pinhole Camera Model can be used [65]. Equation 3.3 shows this model and how it is possible to calculate the distance. In the equation the distance to the object is denoted d, F_c is the focal length of the camera, H_α is the real height of the object, and h_α is the height of the object in pixels. Figure 3.6 shows the general model of the pinhole model.

$$d = F_c * \frac{H_\alpha}{h_\alpha} \tag{3.3}$$

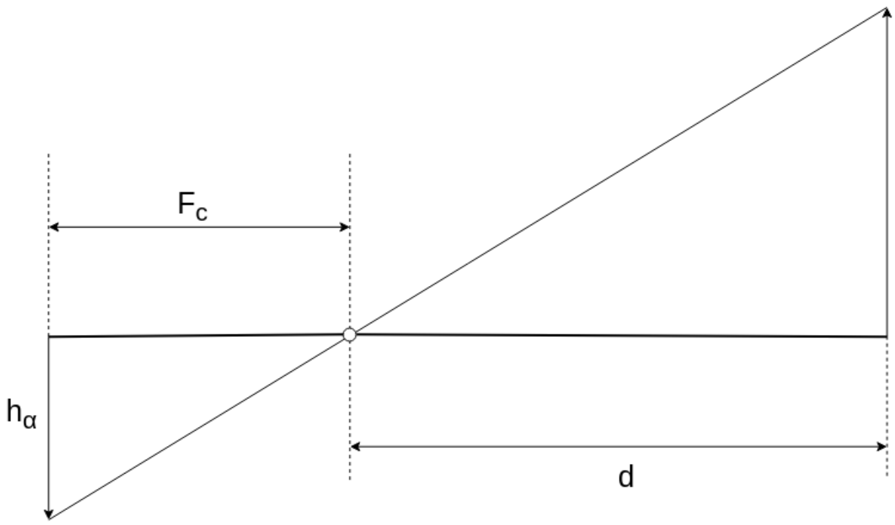


Figure 3.6: The pinhole camera model. F_c shows the focal length, H_α is the real height of the object, h_α is the height of the object in pixels and d is the distance from the pinhole to the object

3.4.2 Using Camera to Estimate Distance to Vehicles

Measuring distance to vehicles in a video can be challenging if the dimensions of the vehicle are unknown. As Equation 3.3 states, the size of the object in real life must be known in order to calculate the distance. Most of the research done on this matter use the Pinhole Camera Model. Han et al. wrote in their paper about how they could measure distance and features of other vehicles using only a monocular camera [66] . The distance were used to control different safety systems in the vehicle, such as forward collision warning (FCW) and autonomous emergency braking (AEB).

The main concern Han et al. experienced was to know the size of the vehicle they tried to calculate the distance to. They thought the most intuitive thing was to use the height of the vehicles to calculate distance, as there were often fewer variations in vehicle height than in vehicle width. However, they found that the use of width was more accurate since it was hard to calculate the exact vehicle height due to factors such as the horizontal line changing. This line would change based on the conditions the vehicles were driving in and would change the height of the vehicles even though they were located at the same distance.

The algorithm proposed by Han et al. depended on the calculation of the width of the vehicle to work. To calculate the width, they provided two methods. The first one was based on lane width as a reference point and only worked if the lane markings were good enough to be detected. This method was the most efficient, and they did a validity check on the lane quality before determining if the quality was over a threshold. If the quality was good enough, they could store the width of the vehicle and use it until the vehicle disappeared from the video. Equation 3.4 describes how the width of the vehicle was calculated if the lane markings were good enough. In the Equation, ω_l is the lane image width and $W_l(k)$ is the physical width of the lane [66].

$$W_v(k) = \frac{\omega_v(k)}{\omega_l(k)} W_l(k) \tag{3.4}$$

To calculate features such as position, velocity, and acceleration, they used Kalman filters [66, 67] with a constant acceleration model. If the lane information was not valid, they used a temporary calculation which lasted until valid lane information was found. Here they used a temporary estimation of the horizontal line and the bottom edge of the vehicle. They used other vehicles' width and horizontal line to estimate each vehicle recursively. Finally, the values were as in the preferred method used as an input in a Kalman filter, and they could calculate the position, velocity, and acceleration.

The results of this algorithm showed that it could be comparable with some of the state-of-the-art algorithms [66]. However, the algorithm had a lower accuracy when the lane information was invalid.

A similar approach to this was the solution developed by Park and Hwang in 2014 [68]. The primary purpose of their research was to provide a method to estimate range for use in vision-based collision warning. They proposed a method to estimate the distance to other vehicles by using the height from the horizon to the bottom of the vehicle. This assumed that the camera pitch angle was zero. If the camera pitch angle was nonzero, the angle also had to be taken into the equation. The reason was that small variations in the horizon position could cause a large error.

Their main contribution was to propose a method that estimated a virtual horizon line, which in turn were used to estimate the range to the vehicle in front [68]. The virtual horizon line was estimated by finding the average horizon height from multiple detected vehicles. Their estimated horizon height could vary because of false detection and an insufficient number of detections. To combat this, they suggested using average height over multiple frames. In case they only detect one vehicle, their method used only size information to calculate the distance with the pinhole camera method.

The object detection part of their algorithm was supported by Haar-like features and AdaBoost. During their experiment, the reference horizon was identified manually. They compared the proposed distance estimation of their method, with the estimated distance found by using the pinhole method. Evaluation of their range estimation demonstrated that each estimation method had good accuracy when the vehicle was within 50 meters. Their error increased when the distance increased above 50 meters.

Another approach was developed by Joglekar et al. in 2011 [69]. They used the geometry and point of contact with the vehicle to calculate a reference point. They mounted the camera in such a way that the optical axis was parallel to the road. By doing this, they could use the height of the objects to calculate the distance by utilizing the similarity of triangles. This algorithm assumed that the road was planar and would not work correctly if this was not the case. Their results showed an accuracy of 96%.

3.4.3 LIDAR Technology

LIght Detection And Ranging (LIDAR) technology is a technology that is useful for generating detailed information about the surroundings. A LIDAR unit consists of several lasers that send and receives light. By rotating the sensor, it is possible to create a 360°field view of the surroundings, and let a vehicle traveling in up to 65 km/h to detect objects as small as 15 cm with a range of 50m. LIDAR sensors can collect 1.3 million data points each second. A LIDAR generates a point cloud of the area based on how the light is reflected, and since the traveling time is known, this can be used to calculate the distance between the sensor and the object [70].

A LIDAR for 3D mapping is expensive and there has been developed different LIDARs that map the environment i 2D instead [71]. Catapang and Ramos tried to develop an object detector that could measure how far away different obstacles were by using a 2D LIDAR in 2016 [71]. The LIDAR they used has a limited range of 40 meters, but their results showed that it was possible to receive good results with an error of only a few centimeters when measuring distance within the LIDARs capable range. Huang and Barth combined CV with LIDAR in 2009 to create an object detection algorithm [72]. They used AdaBoost as their detection algorithm, and the results showed that this combination worked well as they managed to both detect the objects and the distances to them.

3.4.4 Distance Estimation with Stereoscopic Cameras

Stereoscopic cameras are cameras that can give a 3D representation of the image by using two pictures taken at two different positions [73]. This makes it possible to, for instance extract information about distance by knowing the distance between the cameras.

The use of stereoscopic cameras are versatile due to the 3D representation, and they can be used in traffic to estimate distances [74]. Results on the research of this matter show an error rate of around 7% when measuring the distance [74].

3.5 Speed estimation

The equation for calculating speed is straightforward. From basic physics, it is given that speed is the change in position over the change in time. Equation 3.5 shows how this is done.

$$v = \frac{\Delta s}{\Delta t} \tag{3.5}$$

When calculating the speed of vehicles with only a monocular camera, the same equation could be applied. The principle is the same. The following section will make a summary of some research done on this topic.

3.5.1 Measuring Speed from a Monocular Camera Mounted in a Vehicle

Liu et al. demonstrated the estimation of speed based on object detection with YOLO, together with an Optical Flow Calculation [75]. Their approach combined the two CNNs YOLOv2 and FlowNet to estimate the speed of objects in real time. YOLO provided the detection of object size, type, and location, while FlowNet provided the optical flow of the whole image. The object location and size were needed to select the object parts from the optical flow image, to be able to calculate the optical flow for each object. They concluded that their method worked, and they were able to estimate the right speed of an object in real time.

As mentioned in Section 3.4.2, Han et al. measured the distance to other vehicles by using a monocular camera and the width of the vehicle in front [66]. They assumed that the vehicle in front was moving with constant acceleration and used Kalman filters to estimate the speed. As described in Section 3.2.3, Chang and Wong used a stationary camera but utilized the knowledge of the frame rate to keep track of the change in distance per time [5].

Rohit Sharma published an article in January 2019 where a YOLO object detector was used to estimate vehicle speed [76]. The center y-coordinate of the bounding box from YOLO was used as the center point in the speed calculation. His approach was to generate a histogram of all the center-coordinate values and calculated the average speed between the frames. With the center-pixel distance for one frame being calibrated according to the known speed limit of the road, the speed calculation was done by mapping speed of bounding boxes from pixels/sec, to vehicle speed in km/h.

Wu et al. wrote an article in 2015 about how they could calculate the speed of vehicles by only using a stationary monocular camera [64]. The research received great results with 95% accuracy when the ground truth was calculated by a LIDAR.

The first step of their method was to detect the vehicle. They did not track the actual vehicle, but the license plate mounted aft on the vehicle. They used the license plate to detect an initial speed and then calculated the actual height of the license plate to refine the speed. A license plate had consistent features, and this was something that lacked in other vehicle parts. To detect the actual license plate, they limited the ROI by tracking the motion of objects to reduce the interesting objects in the video. This also prevented similar objects such as posters to be detected as license plates. The detection was based on a standard window-search, which was done inside each ROI. After the license plate was detected, they tracked the plate/vehicle by looking at the offset between frame 1 and 2, and then assumed the same offset in the rest of the frames.

As stated in the articles above, the speed could be calculated with reasonably good results, as long as the change in position and a time unit was known.

Chapter 4

Research Design and Motivation

This chapter will present the motivation and strategy for the research.

4.1 Research Motivation

With the focus on autonomous and connected vehicles in the last 5-10 years, and with a focus on better ITS solutions, there were many possible areas for research and development.

Having a traffic environment consisting of only autonomous vehicles would be beneficial with regards to safety and efficiency, but it was not realistically to achieve this in the next couple of years of the time of writing. Until that was the case, there would be a state with mixed traffic containing both autonomous and human-driven vehicles.

Makarem and Gillet demonstrated that knowing the inertia (cost of changing direction and/or speed) and intention of nearby vehicles in an intersection had a great effect on traffic flow and efficiency [77]. They based the simulations on the assumption that all vehicles passing the intersection were autonomous, and could communicate their speed, intention, position, and other information with each other.

As mentioned in Chapter 2, getting all vehicles to communicate with each other, and decide how to pass the intersection the most optimal way, may not be feasible in the nearest future. Another possible solution was to have a system dictate the traffic through an intersection to attempt to get the best possible flow. Each vehicle passing through the intersection would communicate with the management system to provide information about, e.g., speed, path, and intended direction. This could help the ITS decide how and when to pass the intersection the most optimal way for all vehicles in the vicinity.

The motivation for this thesis relied on the potential usage of both existing infrastructure and components in already built vehicles to contribute to improving the traffic flow in signalized intersections.

As stated in Section 2.4, one problem with existing research was the lack of enough vehicles with the right equipment. Relying on expensive equipment, such as LIDARs, could result in not enough vehicles having the equipment installed. By using something cheaper like a camera, the probability of enough vehicles having this installed would be much higher. It could also be cheaper to implement on a scalable basis.

4.2 Research Objectives

We wanted to close the gap between now and the future, where all vehicles were able to communicate their position, speed, and intention when approaching an intersection. We based our research on utilizing equipment that already existed in autonomous vehicles, to investigate the option to use this equipment to extract information about other non-autonomous vehicles. The advantage of focusing on the camera sensor was the possibility cameras had of being installed on vehicles that did not have this as a standard. The main objective of this thesis was, therefore, to investigate if monocular cameras could be used to extract information about the surrounding traffic, which would be communicated to an ITS. An Intersection Manager requires the information described in 2.2, and our objective was to try to provide this information from the extracted information from the camera.

The last objective of this thesis was to provide data for use in further research at the department.

4.3 Research Questions

We developed the following research questions to help us achieve the research objectives of this thesis.

RQ1: How can the number and type of vehicles in nearby lanes of a vehicle be found using image data captured from a vehicle-mounted monocular camera?

RQ2: What is the most accurate combination of width and height when calculating distance using image data captured from a vehicle-mounted monocular camera?

RQ3: How can speed of the nearby vehicles be estimated by using distance calculated from image data captured by a vehicle-mounted monocular camera?

RQ1 was about finding where the vehicles in the vicinity of the vehicle were located. As Table 2.2 stated, the arrival lane of a vehicle was essential for an intersection manager to know. The number of vehicles approaching and the type of vehicle was also required information.

RQ2 was built upon earlier work done in other research. There were many different approaches when calculating the distance to a vehicle, but there was often a choice between using the height or width of the vehicle. The goal of this research question was to see if the results became better when using a combination of these values and which ratio that gave the best accuracy.

Finding the speed of other vehicles was often done by using an algorithm that looked at the change of pixels. RQ3 was about using the change of a distance already calculated in RQ2 and known data from a standard vehicle to calculate the speed and how this could be done with acceptable results.

4.4 Research Method

The research strategy for this project was the Design Science Research Process, described by Peffers et al. [2]. It was chosen because it fits well with this type of project since the desired output is not only academic results but also a working system for further use and development.

The research strategy consisted of six steps, which was answered throughout the thesis. The following list describes the steps and what part of the thesis they relate to.

1. Problem identification and motivation - This step was about defining the problem and why this was a problem to solve. In this thesis, this was located in Section 4.1, where the research motivation was described.

2. Objectives of a solution - The second step was about the objectives of the proposed solution. Should the solution be better than an existing solution, or should it be something new? In this thesis, the objectives for the proposed research results are presented in Section 4.2.

3. Design and development - This step is all about the design and development process of the solution. Chapter 5 describes the full implementation of the system and why different technologies and algorithms were chosen when the system was developed.

4. Demonstration - This is about how the solution performed and how the solution could be used to solve the problem. Chapter 6 describes the result and evaluation of the solution.

5. Evaluation - The results need to be evaluated with the objectives described in step 2. This step was also presented in chapter 6, where each result was evaluated with related work and ground truth if possible.

6. Communication - The final step was about how the results, the problem, and the solution could be communicated to fellow researches and others. This was described more in chapter 7 where the results are discussed.

Chapter 5

Research Implementation

We decided to build a system based on different state-of-the-art algorithms to answer our Research Questions. The system was developed in Python and utilized popular frameworks that had well-documented results in different projects. This chapter presents the different choices made during the research and development of the system for answering the Research Questions. The chapter is divided into different sections where design choices for each research question are presented. Lastly, the whole system pipeline is described.

5.1 Research Question 1

Research Question 1 was about ”How can the number and type of vehicles in nearby lanes of a vehicle be found using image data captured from a vehicle-mounted monocular camera?”.

To answer this question, the system had to accomplish two requirements. The first requirement was the ability to detect other vehicles at the road in the vicinity of the vehicle. The other requirement was the ability to distinguish the vehicles in separate lanes.

5.1.1 Vehicle Detection

Object detection was one of the most resource demanding tasks of a system that are supposed to read the traffic and keep track of vehicles. As written in Section 3.2, there was multiple algorithms and other research done on the subject.

Since the state of the art was good and was fully capable of detecting vehicles, we decided to use an already trained network which had proved great performance on different data sets [46]. As described in Section 3.2, YOLO was a state-of-the-art algorithm that was capable of detecting objects fast and accurate. We chose to use an existing implementation of an object detector with YOLO as a foundation for the system, and further develop this to fit our objectives [78]. The selected implementation was trained on the COCO data set [46, 79].

As the creators of the YOLO algorithm, Redmon and Farhadi stated in their paper about the new version of YOLO, the algorithm runs significantly faster, and the results are more accurate than other similar algorithms such as RetinaNet [46, 80].

5.1.2 Lane Detection

In order to fulfill the second requirement of distinguishing vehicles in different lanes, the lanes had to be detected.

Lane Detection Method

In Section 3.3, we described different methods of detecting lanes used in research. There were several available algorithms, and the main arguments when deciding which one to use was accuracy and speed. After looking at different solutions, we decided to use Progressive Probabilistic Hough Transform since it provided great results for a small cost of computing power [63].

Edge Detection Method

To be able to use the HT approach, the edges of the image had to be detected. As described in Section 3.3.1, there are different methods of doing this. As stated in Section 3.3, almost all used Canny Edge detection [56]. Since it was not too time-consuming to implement all these edge detection techniques, we tried them all to see which one that performed best. Figure 5.1 show the results, and this shows that Sobel Edge has too much noise, while Prewitt showed to few edges. Canny, however, showed a good amount of lane edges, but without much noise. Due to these results and the wide use of Canny in different research, we chose this algorithm.

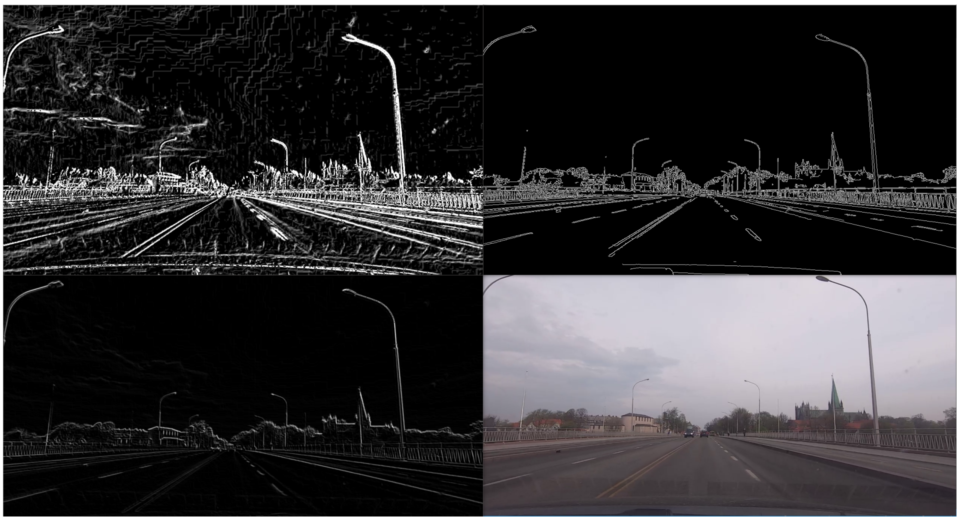


Figure 5.1: Comparison of different edge detection algorithms. Top Left: Sobel Edge Detection, Top Right: Canny Edge Detection, Bottom Left: Prewitt Edge Detection and Bottom Right: The Original Image

Lane Detection Algorithm

To further reduce the computation time, the system converts the image to grayscale to get an intensity value instead of RGB-colors, and also use the ROI approach described in Section 3.3.

The algorithm is shown in Figure 5.2 and was as follows:

1. Convert the image to gray-scale for easier processing by changing colors to intensity values.

2. Blur the image to remove noise before edge detection.

3. Apply the Canny Edge detection algorithm to find edges. The image now consists of only 0's and 1's. Edges are marked as 1's, while the rest of the image is 0. Figure 5.3 show how this looks.

4. Crop the image to remove areas that are not containing lane lines. The ROI was cropped in a way that caused the part to formed as a

trapezoid at the bottom half of the image. Figure 5.4 displays this step.

5. Apply Progressive Probabilistic Hough Transform to find continuous lines in the ROI. This step also divides the line in left and right groups based on the direction of their slope. A positive slope means the lanes go upward from left to right, and the opposite is true for the other lane. Figure 5.5 shows the results of this step.

6. Merge the detected lines to form one long continuous line for each side. Figure 5.6 shows the lanes that the system detected.

The output of this algorithm was image coordinates for the two lane lines.

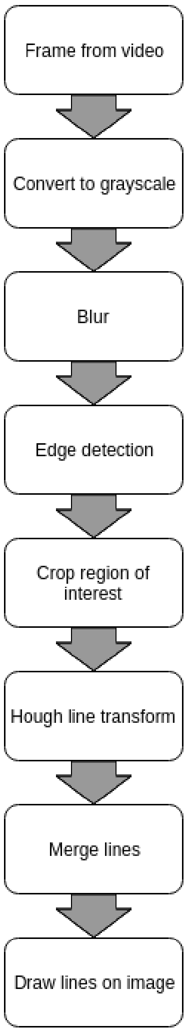


Figure 5.2: Steps of the Lane Detection Algorithm

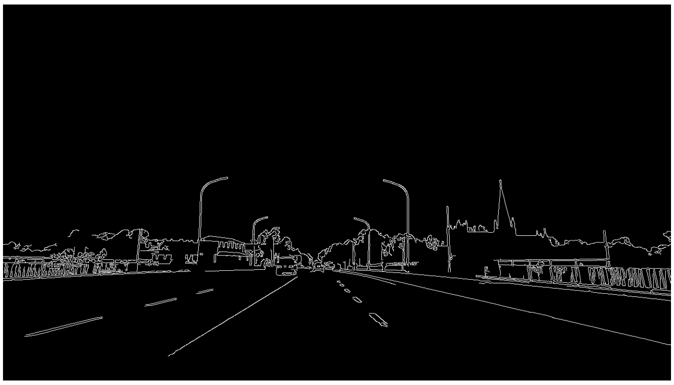


Figure 5.3: Canny Edge Detection

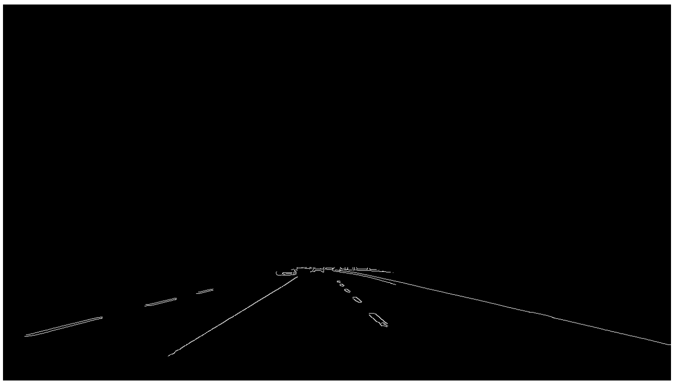


Figure 5.4: The Cropping of the Image Before Hough Transform

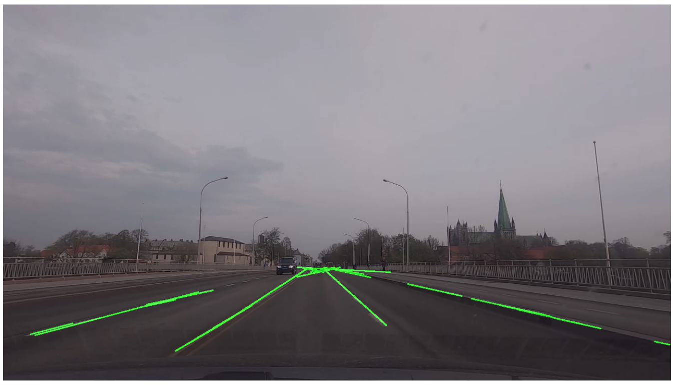


Figure 5.5: Hough Transform Applied

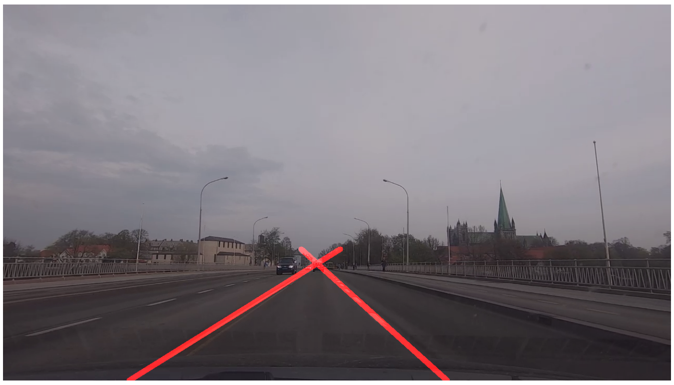


Figure 5.6: Lanes Detected at the Road

5.2 Research Question 2

Research Question 2 was about ”What is the most accurate combination of width and height when calculating distance using image data captured from a vehicle-mounted monocular camera.”

Another important part of the system was knowing approximately where the vehicles were located. Finding the distance was a key component to do this. To use the pinhole camera model find the distance in an image, it was required to know the true size of the object. Research shows that there were different ways of doing this. We wanted to investigate whether using height or width had an impact in accuracy and if combining them gave better results. Moreover, if that was the case, what was the best ratio?

To be able to answer this question, we had to implement a solution that found the distance to a vehicle and do some experiments.

5.2.1 Detection of Distance

As stated in Section 3.4, the pinhole model could be used to calculate distances in images as long as the objects real size was known. This was also the model which were used in other research in the field.

The biggest challenge when calculating distance was often to know the size of the vehicle that should be measured. Other research use reference points such as lane markings [66], or the horizontal line with the vehicles height [68]. Since our research was not safety critical, but should merely provide ancillary information, we looked at combining height and width calculated by using the size of the object from the object detection algorithm. To be able to use the pinhole model for calculating distance, the system had to know the true values for different vehicle types. Table 5.1 show the values used for the calculations, based on approximated sizes of vehicles.

The system calculated the distance twice, first by using height, then by using width. Afterward, the average was computed by using a weight factor to determine the ratio between height and width. Equation 5.1 show how the distance was calculated, based on the distance found for both height and width with Equation 3.3, and how the weight factor was used in the calculation. The factor used to control which of the height or width values that should be prioritized was denoted γ. Calculated distance by height was denoted d_h, and calculated the distance by width was denoted d_w. d_v

Type of vehicle	**Width**	**Height**
Bus	2.4 m	4.0 m
Car	1.8 m	1.6 m
Motorbike	1.0 m	1.0 m
Truck	2.4 m	4.0 m
Van	1.9 m	2.5 m

Table 5.1: Height and width values used by the system for different vehicle types

represented the estimated distance from the camera to the object.

$$d_v = \frac{(1+\gamma)\cdot d_h + (1-\gamma)\cdot d_w}{2} \qquad -1 \leq \gamma \leq 1 \tag{5.1}$$

5.2.2 Distance Estimation Algorithm

Figure 5.7 display the algorithm of the distance estimation part of the system. The algorithm to computed the distance to a detected vehicle was as follows:

1. The detection part of the system provide the type, together with height and width in pixels.
2. Use the type to find the correct, true values of the object.
3. Compute the distance for both width and height.
4. Use the weight factor in the calculation of the average of the two values.
5. Return the estimated distance.

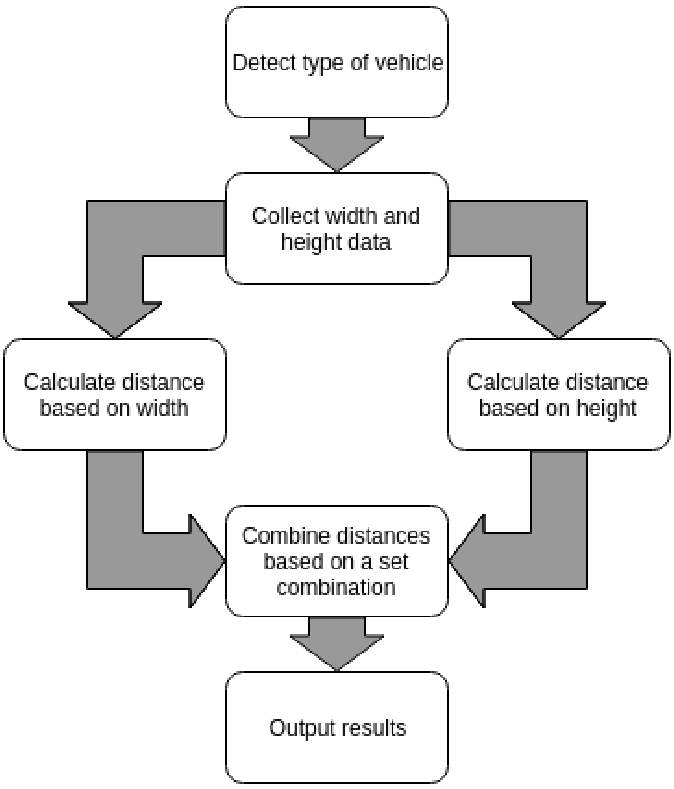

Detect type of vehicle
Collect width and height data
Calculate distance based on width
Calculate distance based on height
Combine distances based on a set combination
Output results

Figure 5.7: Distance Estimation Algorithm

5.3 Research Question 3

The final research question was, ”How can speed of the nearby vehicles be estimated by using distance calculated from image data captured by a vehicle-mounted monocular camera.”. This RQ built upon the other two RQs.

5.3.1 Measuring Speed

As Chang and Wong [5] did when they calculated the speed of the vehicles in the intersections, we chose to use the known frame rate when calculating the estimated speed on the vehicles in the vicinity. Since the FPS specifications of the camera was known, the time between each frame could be used directly.

The intuition behind the method to estimate speed was based on the Equation 3.5. We knew the FPS the videos were captured in and used this to know the time between each frame. 30 frames equaled 1 second, which by Equation 3.5 meant that the change in distance over 30 frames equaled change in distance per time, which was the speed. From the distance estimation part, we had the distance to other vehicles and used that to calculate how much the distance changed. For each vehicle, the system calculated the average change in distance over the last 30 frames to find the change per second. The average was used to remove spikes or other sudden changes in distance, which would cause the speed estimate to spike as well. Therefore, by using the estimated distance to a vehicle, and how much that changed over 30 frames, the system found the speed of each vehicle, relative to the camera. By finally adding the camera’s speed, which was extracted from the GPS data, the speed of each vehicle was found.

5.3.2 Tracking of Vehicles

To be able to calculate the speed of vehicles, we needed to track the vehicles between multiple frames. Since our camera was vehicle-mounted, the same vehicles could be in view for an extended period of time.

To track the vehicles over time, the system calculated and used the centroids of the bounding boxes that the YOLO-implementation created around the vehicles. The system kept track of the centroids between the frames and then used Euclidean distance to calculate the positional difference. On the next frame, the calculated centroids were compared to the previous ones, and the closest centroid in the euclidean distance was considered to be of the same object. This information was then used to calculate the change and then speed of the vehicles. Figure 5.8 show the steps of how the tracking was done.

5.3.3 Speed Estimation Algorithm

Figure 5.9 shows how the system calculated the speed of the vehicles. The system was designed to calculate the speed for all detected vehicles per frame it processed. The steps taken to calculate the speed for a vehicle was as follows:

1. Vehicle is detected by the object detection algorithm and given an ID by the object tracker.
2. The distance of the vehicle is calculated and stored with the object ID.
3. For every frame: Take the change in distance, use the stored data for the vehicle, and calculate the change in distance since the last frame.
4. For every frame: Calculate the average change in distance for the last 30 frames.
5. Return the calculated speed based on the distance estimation.

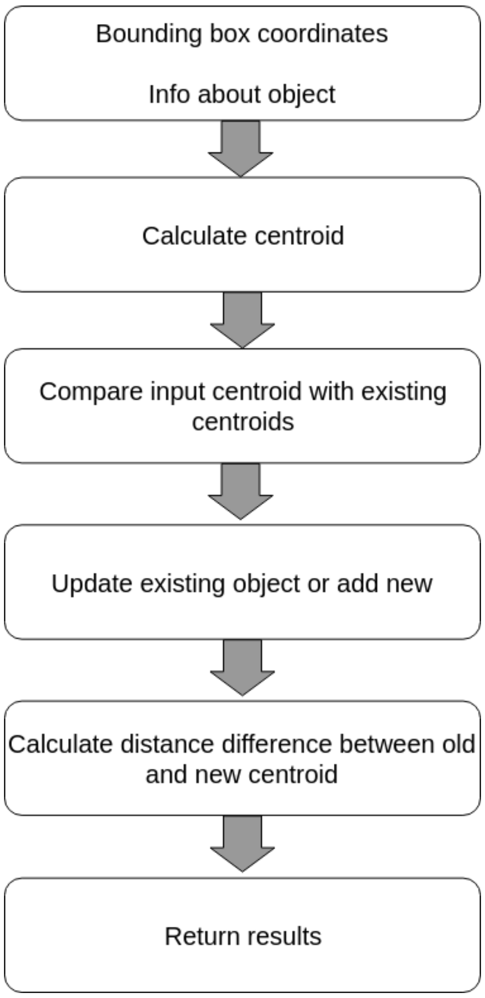

Bounding box coordinates
Info about object
Calculate centroid
Compare input centroid with existing centroids
Update existing object or add new
Calculate distance difference between old and new centroid
Return results

Figure 5.8: Steps of Tracking Objects

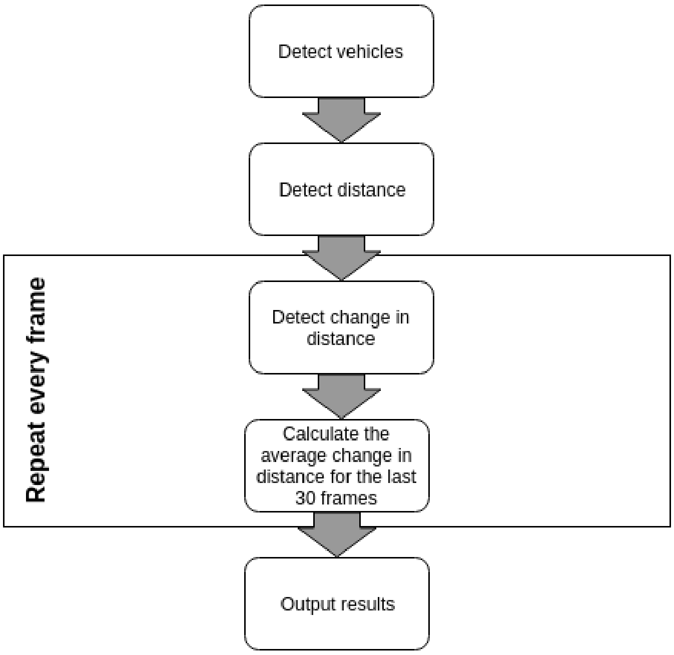

Detect vehicles
Detect distance
Repeat every frame
Detect change in distance
Calculate the average change in distance for the last 30 frames
Output results

Figure 5.9: Speed Estimation Algorithm

5.4 General System Implementation

The system developed incorporated all the parts mentioned for each RQ into one system. In this Section, we will go through the different frameworks and technologies used and also the general system pipeline.

5.4.1 Frameworks

The system was built using existing frameworks written in Python. Different frameworks were used in different parts of the system.

PyTorch

The pre-trained version of YOLO that was used was implemented in PyTorch [81]. PyTorch was as Paszke et al. described it in their paper, ”a library designed to enable rapid research on machine learning models” [81]. The framework was open source and made it easy to implement machine learning models.

OpenCV

We used OpenCV to detect the lanes when driving [82]. OpenCV was an open source CV framework that had implementation of multiple CV algorithms. Since OpenCV already had support for all the edge detection algorithms tested in Section 5.1.2, it was an easy choice to use this framework.

5.4.2 System Overview

The system consisted of the parts described earlier in this chapter, and they were used together to find the relevant data needed to detect, calculate and estimate the data needed to answer the RQs, but also produce data for further research. Figure 5.10 list all the steps of the system.

1. First the video file is imported into the system.
2. Each frame is pre-processed to fit in the ANN for the YOLO-algorithm.
3. The frame goes through two separate algorithms. One does the lane detection, and the other is YOLO, which will detect objects.

4. The output from lane detection and YOLO will be used in the vehicle counting part, which will count all vehicles in each lane.
5. Distance to all detected vehicles is calculated using the pinhole model and values from the object detection algorithm.
6. Each vehicle are assigned an ID and stored, or updated if already existing.
7. The speed for each vehicle is estimated by using the stored data from the tracking and the speed from the GPS-data.
8. This process is repeated for each frame in the video.
9. Every 5 seconds the detected and computed info is saved to a file for later use, e.g., analysis or simulation.

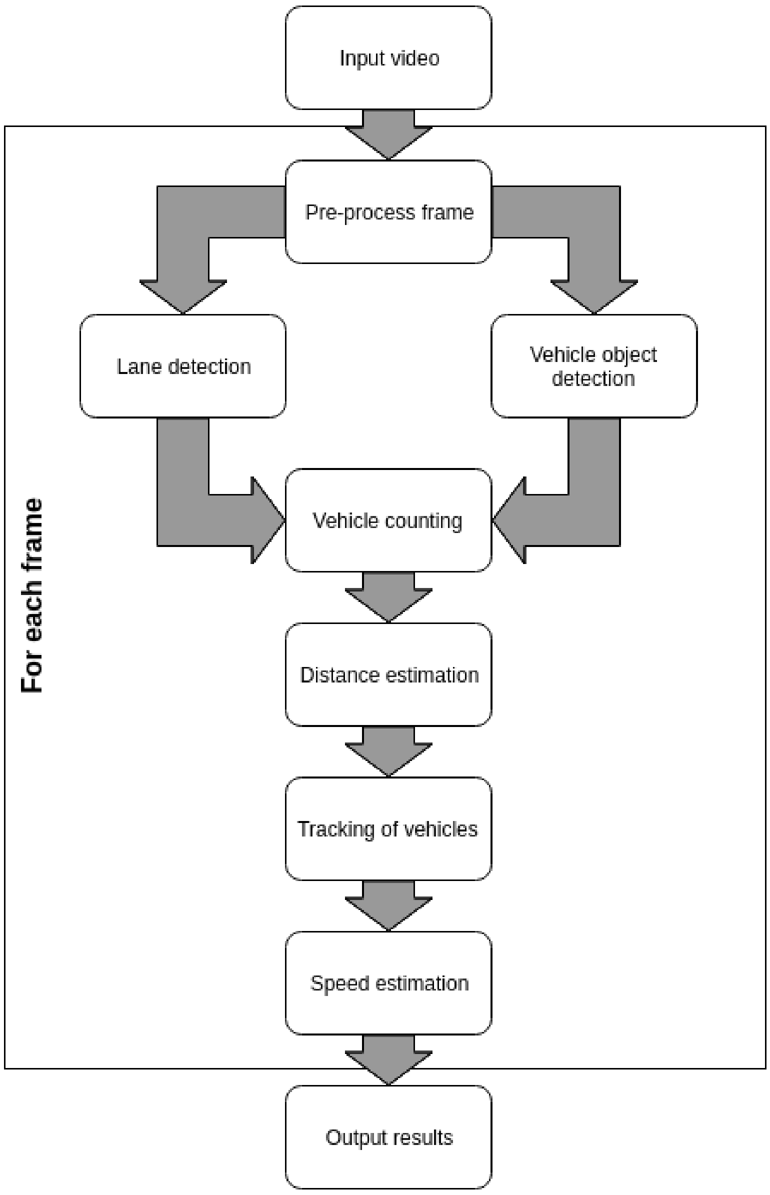


Figure 5.10: System Overview

Chapter 6

Research Results and Evaluation

In this chapter, the results of the research are presented. The chapter is divided into different sections, where the results of each research question are presented and evaluated.

6.1 Data Collection

To be able to do experiments and test our system, we needed video footage of traffic and road. Because we did not have a video an autonomous vehicle, we had to record our own footage. We chose to use a GoPro Hero 7 camera [83] since we had one available and the camera had the ability to record GPS-data. To obtain video footage, we mounted the GoPro to our car and drove a pre-planned route in Trondheim while recording. The video was captured at a resolution of 1920 x 1080, at a frame rate of 30 FPS. Since the GoPro had built-in video stabilization, this was also used. The GPS-sensor recorded data every 55 ms, and each GPS-entry contained latitude, longitude, altitude, speed, and a UTC stamp.

The route we planned had different types of the road such as motorway with multiple lanes, city traffic with traffic lights, busses, and pedestrians, road sections with tunnels or roundabouts, and other mixed traffic. The recording took place between 9-10 AM on a normal workday. Figure 6.1 show the route plotted in Google Maps [84]. After the video was recorded, it was necessary to split the footage into manageable sequences and extract the GPS data into separate files. A simple video editing program was used to split the videos, and an online tool was used to extract GPS-data into a JSON-file [85]. The JSON-file contained a list of GPS readings for the whole video file. The GPS-file would also be split into separate files to match the video sequences.

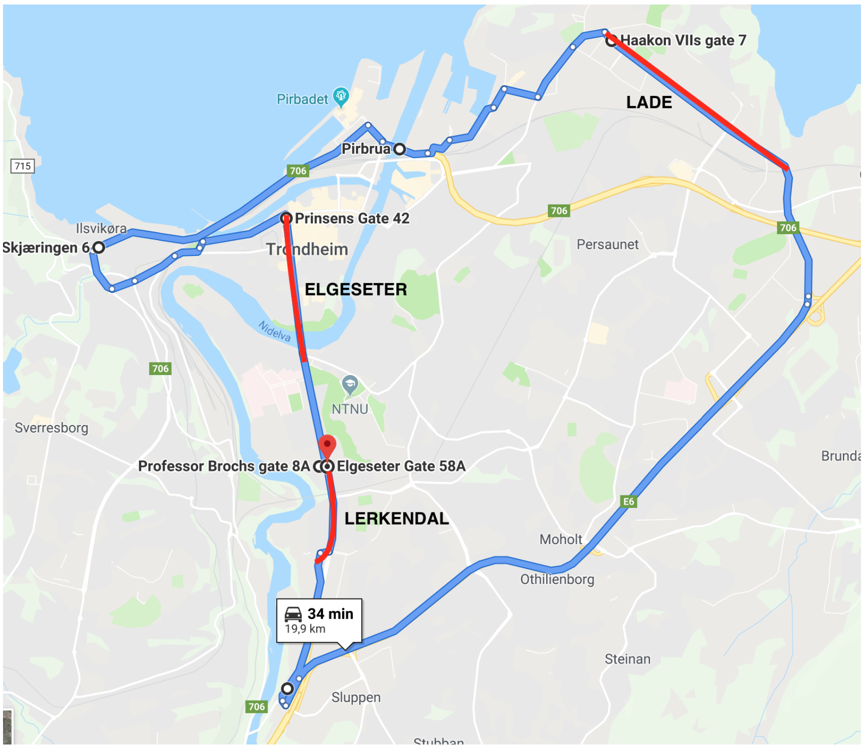
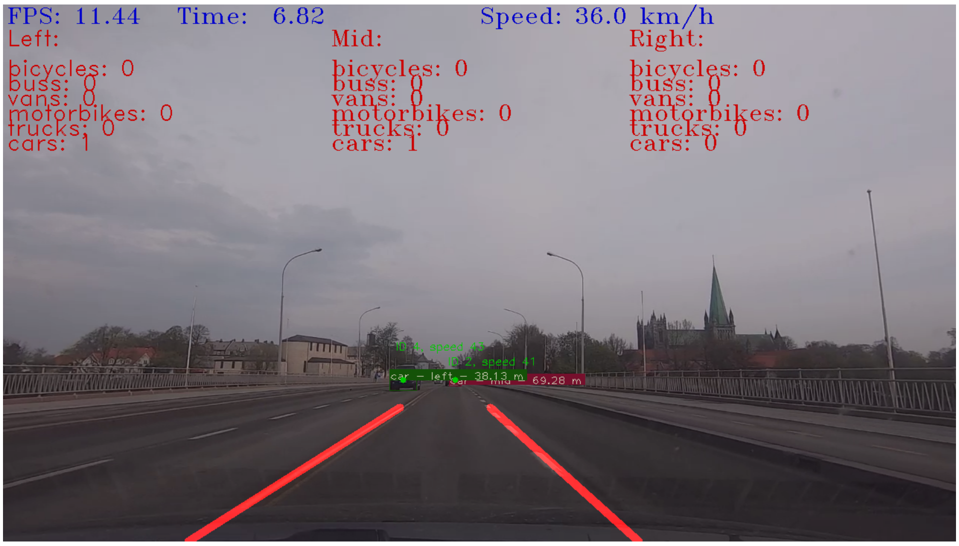


Figure 6.1: Route used to collect data. Google Maps was used to plot the map [84] and the location of different scenarios used are marked in red.

6.2 RQ1 - Finding position and lane counting

The first Research Question was *"How can the number and type of vehicles in nearby lanes of a vehicle be found using image data captured from a vehicle-mounted monocular camera?"* To attempt to answer this question we used a state of the art object detection algorithm, YOLO, together with popular image processing techniques to detect lanes on the road, and built a system to detect vehicles and their relative position. Information from object detection and lane detection were merged and processed to give information about which lane each detected vehicle was positioned in. The individual components of the system are described in Chapter 5. This section will evaluate the performance of the object detection and lane detection part of the system.

Figure 6.2 show how the system displayed information during run time. The system updated number of vehicles every 2-3 seconds, which was the average number of vehicles it detected over the last 2-3 seconds worth of frames (which at 30 FPS were 60-90 frames).



```
FPS: 11.44    Time:  6.82              Speed: 36.0 km/h
Left:                      Mid:                     Right:

bicycles: 0                bicycles: 0              bicycles: 0
buss: 0                    buss: 0                  buss: 0
vans: 0                    vans: 0                  vans: 0
motorbikes: 0              motorbikes: 0            motorbikes: 0
trucks: 0                  trucks: 0                trucks: 0
cars: 1                    cars: 1                  cars: 0
```

Figure 6.2: Lanes and vehicles detected on the road

6.2.1 Testing procedure

To evaluate the performance of the system with respect to RQ1, we took three different sequences of varying duration and location from the route (Figure 6.1) and compared the results from the system with a manually counted number of vehicles for the corresponding sequence. The system output and manually counted vehicles were noted every 2-5 seconds. We have these two performance measures for our tests:

- Measure 1: Overall detection- and counting ability of objects, not respecting vehicle type.
- Measure 2: Number of times vehicles were counted correct and incorrect in different lanes, respecting vehicle type.

The first performance measure was associated with the performance of the object detection part of the system and the ability to identify vehicles. The second performance measure was linked to how well the system was able to correctly identify and count the correct number of vehicles in a lane.

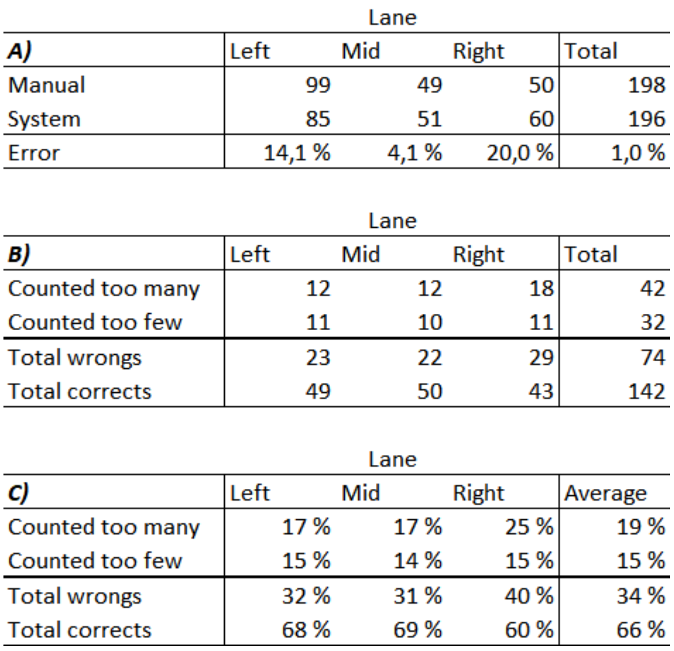
In addition to these performance measures, when a wrong counting occurred, we examined more closely why the system made that mistake. For example, when the counted numbers were wrong, we inspected the visual detection output to examine if the lanes were detected wrong or if the classification of the vehicle caused the error. This gave an indication on which parts of the system were not performing or working as desired and could require some further development.

6.2.2 Results

Scenario 1 - Elgeseter - City traffic

Scenario 1 were captured northbound on Elgeseter street and could be classified as city traffic, with several traffic light intersections, busses and pedestrians. The video sequence was 4 minutes long in total, and 72 readings were done, giving a reading approximately every 3.5 seconds. Table 6.1 show how the system performed with respect to the performance measures.

In the tables, ”counted too many” and ”counted too few” means how many times the system counted too many vehicles or too few vehicles, compared to the true number of vehicles in each of the lanes. Together these two values gives the total amount of wrong countings for each lane.



	Lane			
A)	Left	Mid	Right	Total
Manual	99	49	50	198
System	85	51	60	196
Error	14,1 %	4,1 %	20,0 %	1,0 %

	Lane			
B)	Left	Mid	Right	Total
Counted too many	12	12	18	42
Counted too few	11	10	11	32
Total wrongs	23	22	29	74
Total corrects	49	50	43	142

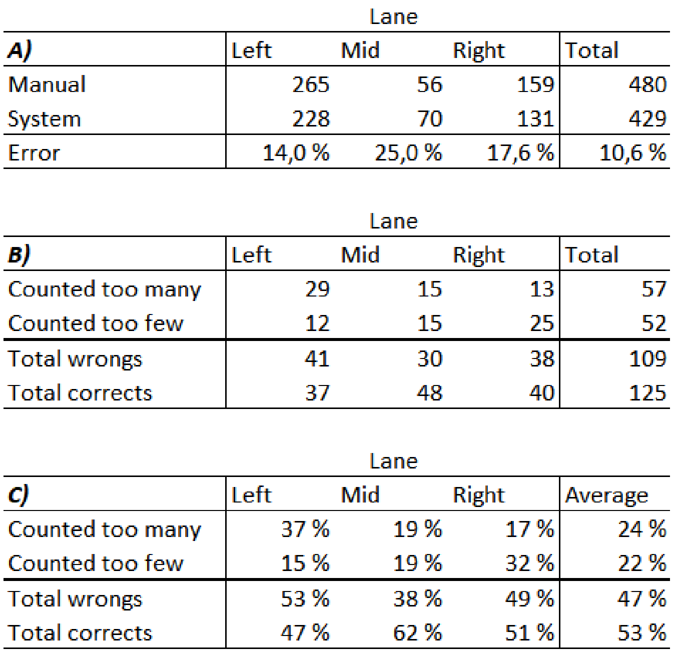
	Lane			
C)	Left	Mid	Right	Average
Counted too many	17 %	17 %	25 %	19 %
Counted too few	15 %	14 %	15 %	15 %
Total wrongs	32 %	31 %	40 %	34 %
Total corrects	68 %	69 %	60 %	66 %

Table 6.1: Results from Scenario 1
A) Total counted objects of the sequence, not respecting object type.
B) How many times the system counted right or wrong out the 72 readings, also respecting object type.
C) The percentage distribution of subtable B)

Scenario 2 - Lade - Mixed traffic

Scenario 2 were captured along Haakon VIIs street at Lade. This video sequence had normal to busy traffic, with multiple traffic light intersections and crossing traffic. A repeating source of error for this sequence was the surrounding parking lots, causing vehicles from the parking lots be detected by the system. The duration was 3.5 minutes, with 78 readings, giving a reading every 2.7 seconds. Table 6.2 show how the system performed with respect to the performance measures.

In the tables, ”counted too many” and ”counted too few” means how many times the system counted too many vehicles or too few vehicles, compared to the true number of vehicles in each of the lanes. Together these two values gives the total amount of wrong countings for each lane.



	Lane			
A)	Left	Mid	Right	Total
Manual	265	56	159	480
System	228	70	131	429
Error	14,0 %	25,0 %	17,6 %	10,6 %

	Lane			
B)	Left	Mid	Right	Total
Counted too many	29	15	13	57
Counted too few	12	15	25	52
Total wrongs	41	30	38	109
Total corrects	37	48	40	125

	Lane			
C)	Left	Mid	Right	Average
Counted too many	37 %	19 %	17 %	24 %
Counted too few	15 %	19 %	32 %	22 %
Total wrongs	53 %	38 %	49 %	47 %
Total corrects	47 %	62 %	51 %	53 %

Table 6.2: Results from Scenario 2
A) Total counted objects of the sequence, not respecting object type.
B) How many times the system counted right or wrong out the 78 readings, also respecting object type.
C) The percentage distribution of subtable B)

Scenario 3 - Tempe to Lerkendal - Mixed traffic

Scenario 3 were captured from Tempe to Lerkendal. This route had normal to busy traffic, and the lanes were separated by a central reservation with a medium-high fence. The duration was approximately 1.5 minutes, with 27 readings, giving a reading every 3.5 seconds. Table 6.3 show how the system performed with respect to the performance measures.

In the tables, ”counted too many” and ”counted too few” means how many times the system counted too many vehicles or too few vehicles, compared to the true number of vehicles in each of the lanes. Together these two values gives the total amount of wrong countings for each lane.

	Lane			
A)	Left	Mid	Right	Total
Manual	35	39	24	98
System	40	21	31	92
Error	14,3 %	46,2 %	29,2 %	6,1 %

	Lane			
B)	Left	Mid	Right	Total
Counted too many	6	8	6	20
Counted too few	5	1	2	8
Total wrongs	11	9	8	28
Total corrects	16	18	19	53

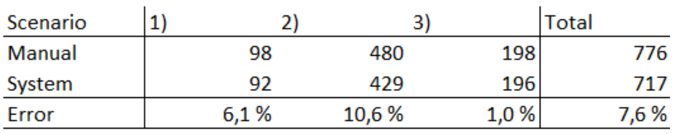
	Lane			
C)	Left	Mid	Right	Average
Counted too many	22 %	30 %	22 %	25 %
Counted too few	19 %	4 %	7 %	10 %
Total wrongs	41 %	33 %	30 %	35 %
Total corrects	59 %	67 %	70 %	65 %

Table 6.3: Results from Scenario 3
A) Total counted objects of the sequence, not respecting object type.
B) How many times the system counted right or wrong out the 27 readings, also respecting object type.
C) The percentage distribution of subtable B)

6.2.3 Evaluation

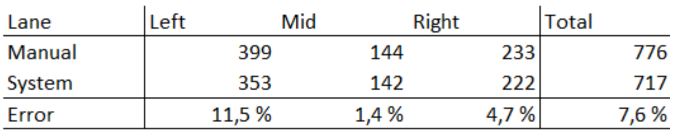
By evaluating the results from each of the three scenarios we got an indication on how the system performed, and how accurate it were able to find the lane position of vehicles. Table 6.4 showed the total error when all readings from all three scenarios were combined. Table 6.5 show the total error when the three scenarios were combined, but for each lane independently. Looking at performance with respect to measure 1, not respecting type, the system was able to detect objects with an overall average error of 7.6%.

Compared to [51] from Section 3.2.3, which achieved an average error of 5.3% for their vehicle counting system, an error of 7.6% was considered acceptable. Considering they used a downward facing stationary camera mounted above the road, they had less potential sources of error with fewer possible interfering objects. Our vehicle-mounted camera was parallel with the road and captured much more of the surroundings. The fact that the camera was moving and did not have a constant background probably caused a greater potential of noise and faulty detections of objects not related to the traffic. Since our numbers, compared with the calculated per-lane error of [51] in Table 6.6, was similar, they confirm that the object detection part of our system performed well when only looking at vehicle detection and not respecting vehicle type.



Scenario	1)	2)	3)	Total
Manual	98	480	198	776
System	92	429	196	717
Error	6,1 %	10,6 %	1,0 %	7,6 %

Table 6.4: Total vehicle detection error for scenarios in Section 6.2



Lane	Left	Mid	Right	Total
Manual	399	144	233	776
System	353	142	222	717
Error	11,5 %	1,4 %	4,7 %	7,6 %

Table 6.5: Total vehicle detection error, per lane, for scenarios in Section 6.2

Lane	Error
1	0,0 %
2	2,8 %
3	10,5 %
4	12,2 %
5	200,0 %
Total	5,3 %

Table 6.6: Calculated accuracy of results from Pancharatnam and Sonnadara [51], based on numbers from Table 3.5

Performance measure 2 were used to assess the systems ability to correctly count and place vehicles in different lanes. To achieve a correct counting, the system had to detect the exact number of vehicles in that lane, also respecting vehicle type. This requirement made this a very strict measure. From Table 6.7 we saw that the average error across all the three scenarios were 38%, which meant that the total number of vehicles across all lanes were counted wrong in 38% of the manually counted measurements. Because counting even one vehicle wrong in any of the lanes counted as an error, the threshold for an error appeared to be low since the failure rate seemed high. Table 6.8 showed that the counting error per lane differed somewhat, with the mid lane being most accurate but only slightly better than average.

Scenario	1)	2)	3)	Average
Total wrongs	35 %	47 %	34 %	38 %
Total corrects	65 %	53 %	66 %	62 %

Table 6.7: Average counting error for the system by scenario

Lane	Left	Mid	Right	Average
Total wrongs	42 %	34 %	40 %	38 %
Total corrects	58 %	66 %	60 %	62 %

Table 6.8: Average counting error for the system by lane

Table 6.9 show that most of the errors were caused by the **Detection** part of the system misclassifying vehicles, and infrequently not detecting the vehicle at all. The other main cause of error was the **Lane markings**. Missing or faulty lane markings on the roads happened frequently and caused vehicles to be counted in the wrong lane. Additional sources were **Distance**, which meant that the vehicles were too far away to get an accurate detection and classification, and **Delay**, which meant the system sometimes did not update the average until vehicles were out of the frame. The last category was **Other vehicles**, which meant other vehicles that were not a part of the traffic. Especially in Scenario 2, this was a major problem since the road was closely surrounded by parking lots. This caused many errors.

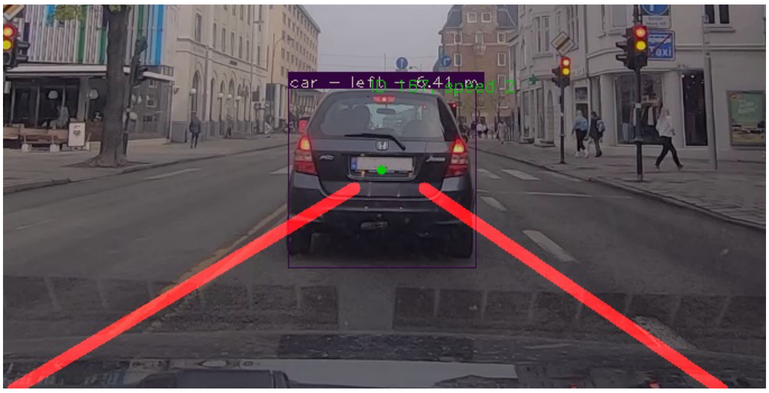
It was not given that each of the sources of error was exclusive. An error could have multiple causes, such as that both erroneously lane markings and a faulty detection contributed to the error. This is why the percentages add up to over 100% in Table 6.9.

	Scenario			
	1)	2)	3)	Average
Number of errors	47	66	19	
Lanes	28 %	39 %	47 %	38 %
Distance	30 %	8 %	5 %	14 %
Detection	62 %	30 %	79 %	57 %
Delay	13 %	0 %	11 %	8 %
Other vehicles	0 %	50 %	0 %	17 %

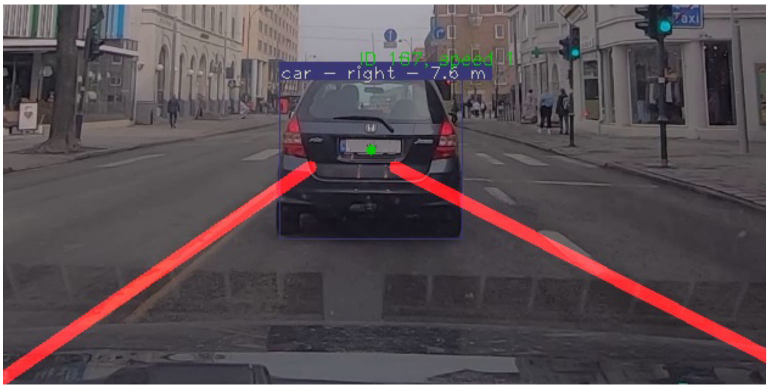
Table 6.9: Sources of error categorized

Most of the errors belonged to the category ”Detection,” which included all errors tied to the vehicle detection part of the system pipeline. This was mostly wrongly classified vehicles, which occurred when for instance a car was classified as a van, or a bus classified as a truck. Often the reason for these misclassifications was objects interfering with the line of sight, overlapping vehicles or the same vehicle being detected multiple times. Only a small portion of the detection errors was caused by objects not being detected at all. Although the object detector was able to detect most of the objects, as mentioned earlier in this section, the vehicles were not always assigned the correct type. This became apparent when evaluating what caused most of the errors.

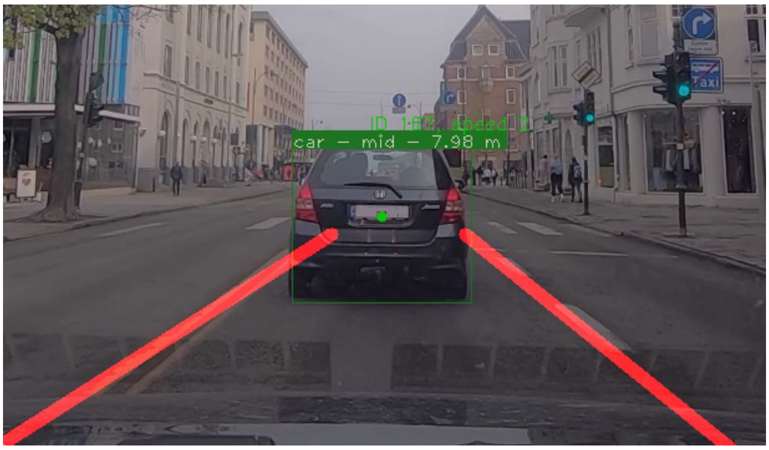
The other relatively major source of error was wrongfully detected lanes or lack of detected lanes. The strict evaluation criteria of performance measure 2 resulted in some errors counting double, which contributed to explain why the error in Table 6.7 and 6.8 were high. For instance, if the algorithm found two vehicles in the center lane but in reality, there was one in the center lane and one right lane, this counted as two errors. This problem occurred if the lanes were detected too far off to one side or the angle was incorrect. Figure 6.3 show an example of how detected lanes with small variations, would result in multiple errors. The vehicle in the figure had three different positions in a short amount of time due to wrong lane detections. The figure shows that the lanes varied somewhat and that this caused the error.



(a) Vehicle detected left



(b) Vehicle detected right



(c) Vehicle detected mid

Figure 6.3: Lane detection faults on the same vehicle

Another problem with lane detection was related to the vanishing point, which corresponds to the Distance category in sources of error. Vanishing point was where the lanes meet and vanish in the horizon. When objects were too far ahead, they became very small, and the system struggled with detecting them. This was a known limitation of YOLO, as mentioned in Section 3.2.2. In addition to this, the lane markings and lane detection lines intersected at the vanishing point, which made it difficult to distinguish which lane a vehicle far away belonged to. Minor variations of the detected lanes cause great variations at the vanishing point. Figure 6.4 show how vehicles far away appear, and that it could be challenging to distinguish which lane they belong to.

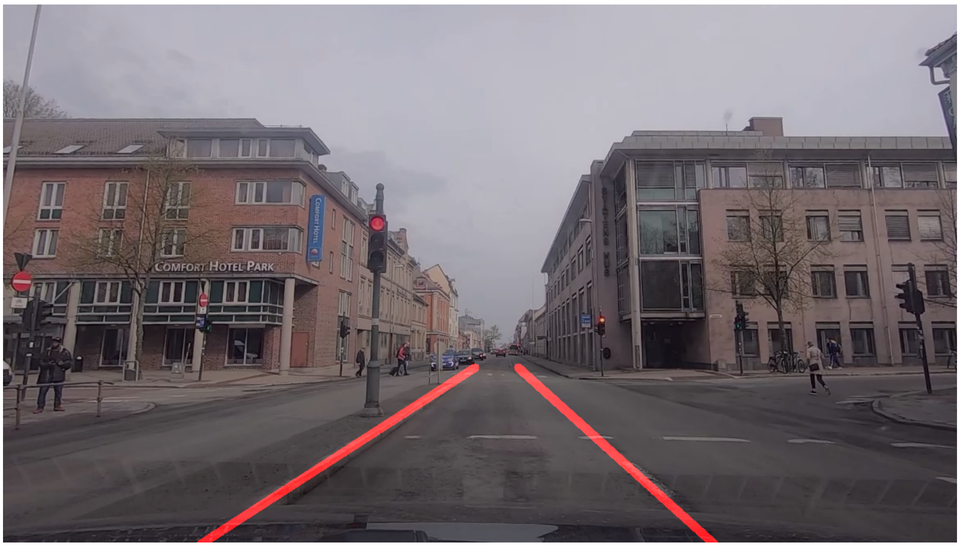


Figure 6.4: Example of vehicles clustered together at the vanishing point

The last problem with lane detection, which caused errors was the problem with bad or non-existing lane markings. If no lane markings were present, or the system found presumable better edges, the lanes could be completely wrong, which caused many errors. Figure 6.5 show an example when the system did not find any lane markings, used a wrongly detected edge which placed the lane line incorrectly, and therefore detected the vehicles erroneously in the center lane. Figure 6.6 show a situation when the system detected a wrong lane marking and why it was detected this way.

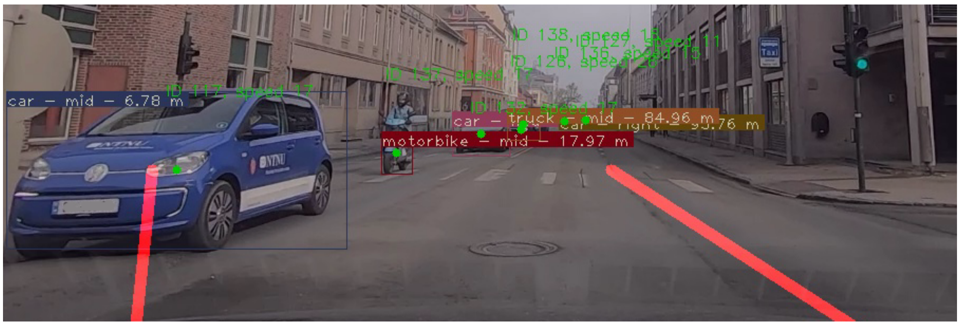
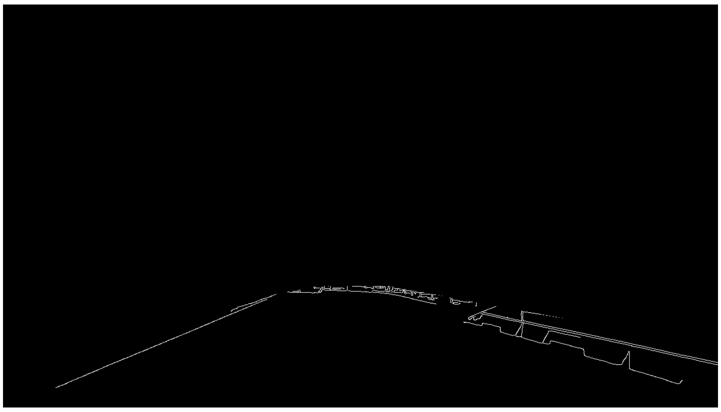
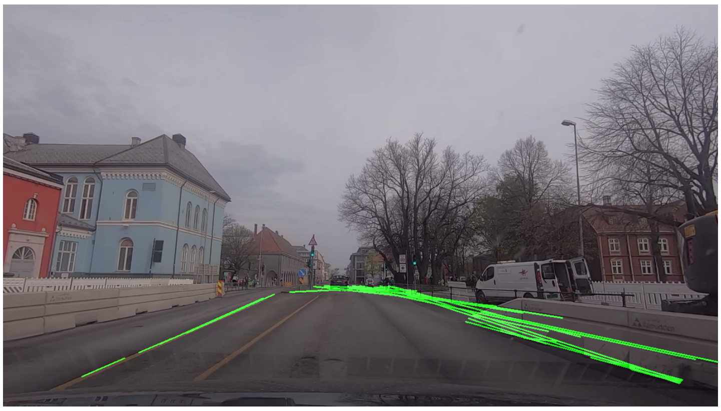


Figure 6.5: Faulty lane detection


(a) Edges found


(b) Detected lines

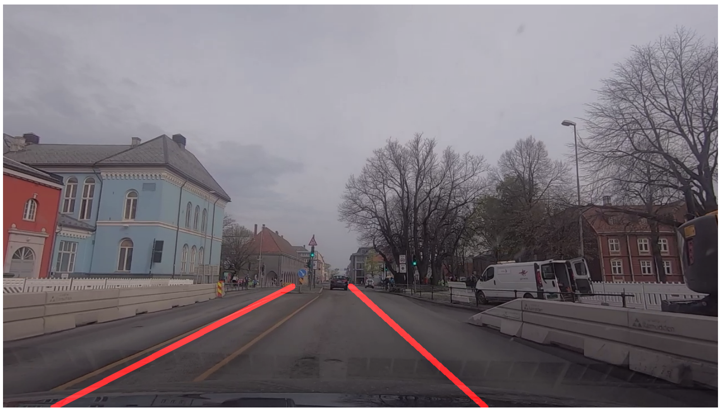

(c) Detected lane lanes

Figure 6.6: Detection of wrong lane markings caused wrong lane line to be detected

6.3 RQ2 - Combination of Height and Width when Calculating Distance

With Research Question 2, we wanted to explore whether using height or width of the vehicle had any impact when calculating the distance to the camera using the pinhole model. Would the distance estimate get more accurate if we used a combination of the two, and in that case, what was the most accurate combination?

6.3.1 Test procedure

The experiment for this part needed its own videos, so these recordings were captured separately from the previously mentioned data collection 6.1. Data for these tests were acquired by recording video clips of stationary cars at different lengths and measure the distance with a simple laser measuring tool[1]. Then the different video clips were run through the system with different ratios of height and width. The output was compared with the truth from the laser to find which ratio was most accurate.

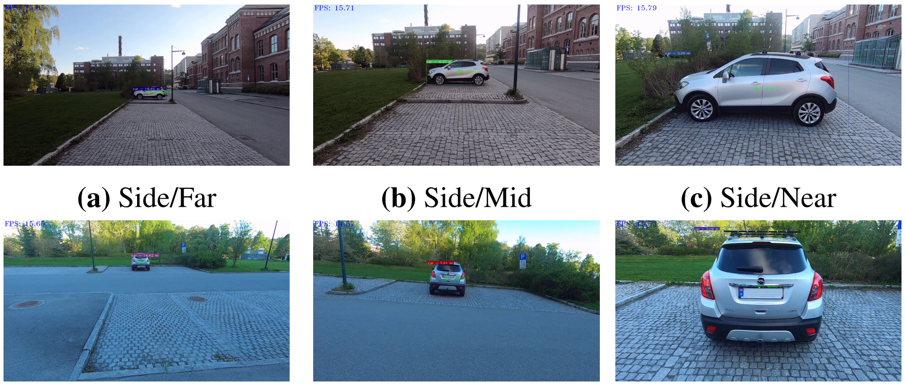
6.3.2 Results

Figure 6.7 show one of the test vehicles measured at different lengths and angles. Figure 6.8 show the average error of different tests with varying ratio of height and width. The best ratio we found was 85% of the height and 15% of the width.

6.3.3 Evaluation

The probable reason the found ratio of 85% height and 15% width gave the best result was that height was the most stable measurement, as shown in Table 6.8. Depending on which angle a vehicle was detected, the width varied more than height. A vehicle detected from the side should not even be measured by the width, but rather by the length. Determining if a vehicle was seen from the behind or the side was out of the scope for this system. In traffic situations, the vehicles would rarely be detected straight from the

[1]This was a simple consumer laser measuring tool, and could unfortunately not be used when driving.



(a) Side/Far **(b)** Side/Mid **(c)** Side/Near

(d) Behind/Far **(e)** Behind/Mid **(f)** Behind/Near

Figure 6.7: Distance to vehicle measured at different lengths and angles

behind, and therefore some of the length would be considered as width. Because the length of a vehicle is longer than the width, the computed distance would have greater fluctuations when some of the length are considered as width, causing the the detected width to be wider than the actual width of the vehicle. Therefore, to achieve a stable estimate, a combination of height and width were used, where the width had lesser weight than the height.

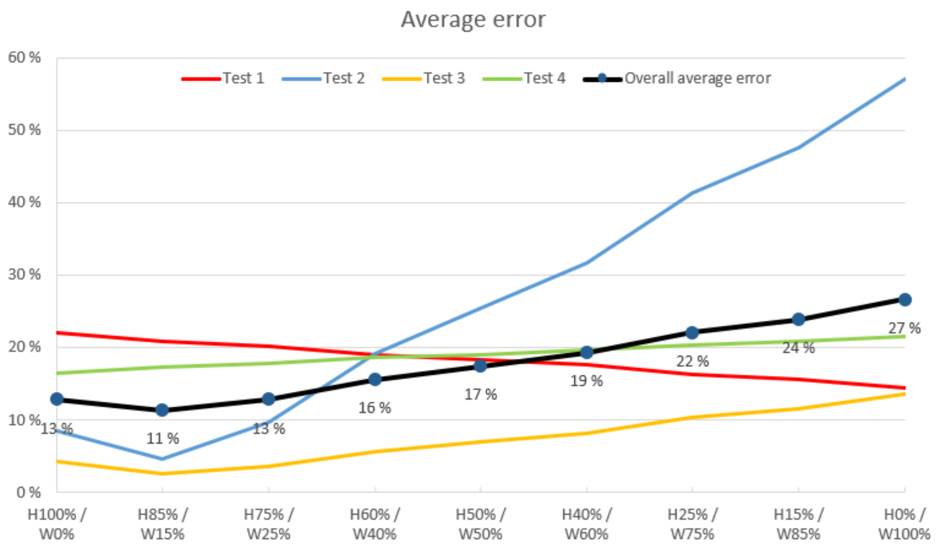


Figure 6.8: Average error with varying ratio of height and width

6.4 RQ3 - Estimate Speed Using Distance

The last Research Question was *”How can speed of the nearby vehicles be estimated by using distance calculated from image data captured by a vehicle-mounted monocular camera”*. To answer this question, we used the already calculated distance and looked at the speed of the filming vehicle and the change in distance over time to do the calculation. The implementation is described more in-depth in chapter 5.

6.4.1 Test procedure

To evaluate the accuracy of the speed estimation, the estimated speed of a vehicle should be compared to its true speed. However, since we did not have the equipment to measure the true speed for vehicles in the recordings, we had to find another way to calculate the accuracy of the systems speed estimation. By assuming the relative distance from the distance estimation was correct, we used traveled distance per time, to calculate the ”true” speed. Since we knew the FPS of the videos, we could accurately calculate the difference in time between x number of frames. Utilizing this knowledge, we manually analyzed the output of the system to get values for comparison.

First, a suitable sequence of output had to be found among different video sequences. A suitable sequence was a sequence where a vehicle was in view for an extended period of time, which meant the vehicle was mostly driving in front of the camera vehicle. Then, at the start of the sequence, we noted the values for our speed, the distance to the designated vehicle, and also its estimated speed. We then proceeded to manually step forward an arbitrary number of frames, mostly in the range of 30-60 frames, and after the given number of frames, we noted the same values, with the addition of noting down how many frames were skipped. This process of stepping a number of frames and noting down output values from the system was repeated until a sufficient amount of data were acquired.

The ”true” speed of the designated vehicle was calculated for each of the selected frames. The method to calculate this speed was to first calculate the time since the previous measurement, then find the traveled distance of the designated vehicle and lastly dividing this distance by the time, which gave the speed. The traveled distance of the designated vehicle was found by taking the distance the camera vehicle moved, calculated by taking speed multiplied with time, adding this with the distance to the selected vehicle, and then subtracting the previously recorded distance to the selected vehicle.

6.4.2 Results

The same videos from the previously mentioned data collection, Section 6.1, were used for these tests. As they also were already run through the system, from looking at RQ1, the same output was used for this analysis.

Figure 6.9, 6.10 and 6.11 show the results of the three manually analyzed video sequences. The graphs show the ”true” speed, based on the calculation described in the test procedure, and the estimated speed done by the system. The average difference of the total 75 manual readings across all the sequences was 2.09 m/s, and the maximum difference was 10.64 m/s.

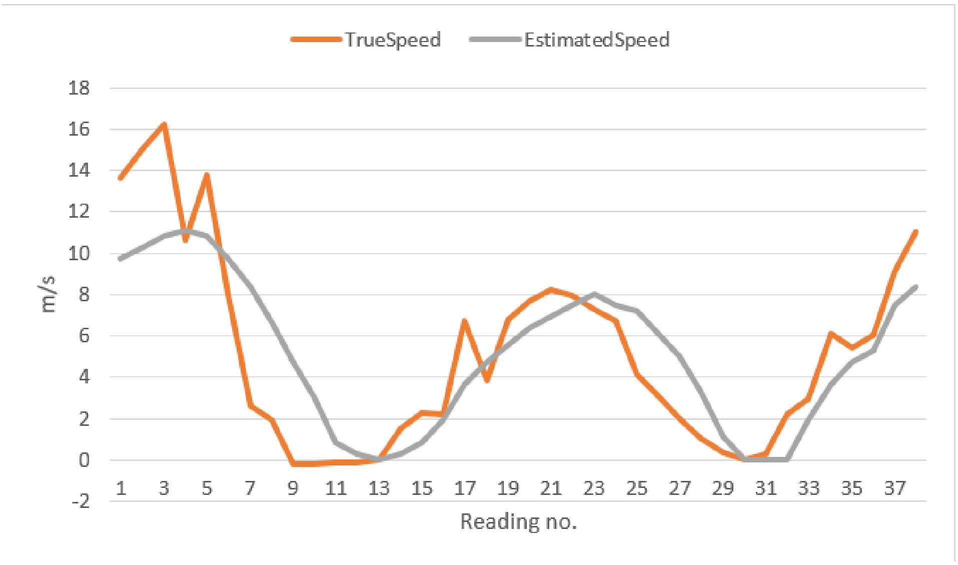


Figure 6.9: Speed analysis 1 - Plot of the true calculated speed and estimated speed of the system

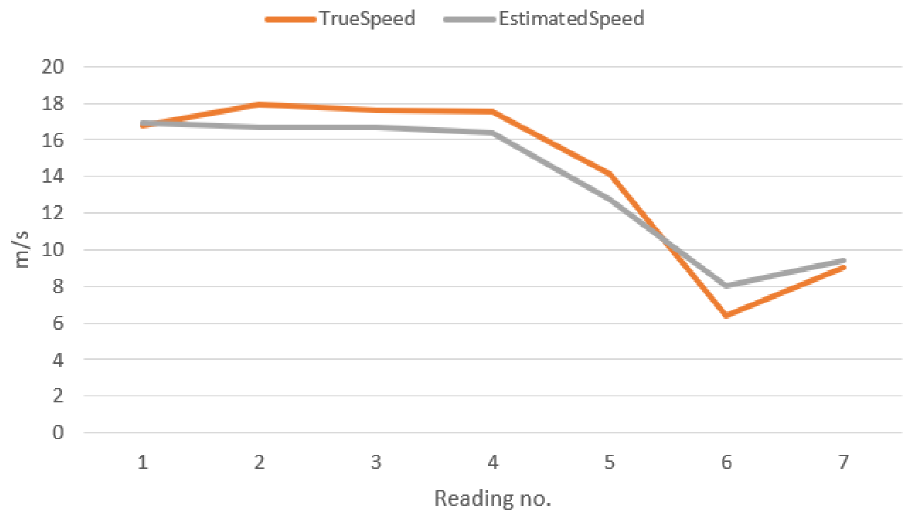


Figure 6.10: Speed analysis 2 - Plot of the true calculated speed and estimated speed of the system

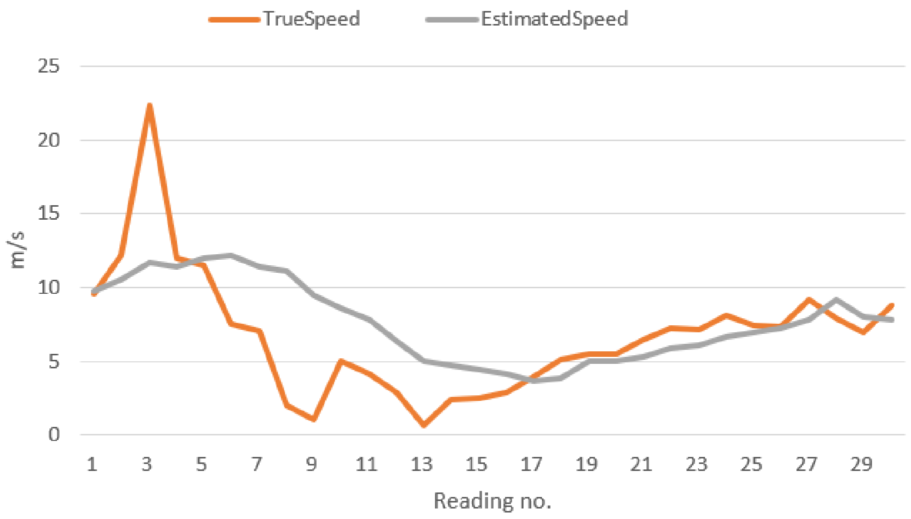


Figure 6.11: Speed analysis 3 - Plot of the true calculated speed and estimated speed of the system

6.4.3 Evaluation

As seen in the graphs, the estimated vehicle speed was not very far off in either scenario. When analyzing the videos, we experienced that estimated speed lagged slightly behind compared to the true speed. It was often the case that if the speed of the vehicle went down, it took a couple of seconds until the estimated speed slowed down correspondingly. Similar if the speed increased. This is best visible on Figure 6.9. The reason was that the speed calculation, which takes the average distance the last 30 frames to calculate the speed, took some time to respond to sudden changes.

Wu et al. [64] managed to receive an accuracy of 95% when they measured speed with a stationary camera and used the license plates as a reference point. Our results were not equally accurate, and the mentioned slight delay in the update of the speed was possibly the leading cause of this. It is, however, complicated to directly compare only the error rate. Since speed is a relative measure, the error rate would be significant with errors on low speeds. For example, our biggest error was 2300% in one of the readings. The reason was that the calculated ”true” speed was very low, below 1 m/s, and the estimated speed was a couple of meters per second. Using this in an overall calculation of the error rate would have a dramatic impact on the result. We, therefore, chose to present our accuracy in terms of average and maximum difference between estimated and ”true” speed for all the three sequences.

Chapter 7

Discussion

This chapter will discuss the different results and limitations with the research of this thesis. The chapter is structured with a discussion of each research question before a discussion of the general system at the end.

7.1 Finding Position and Lane Counting

This section will discuss Research Question 1. The results of this question were presented and evaluated in Section 6.2. The results proved that the system developed to answer the questions had some limitations, but also solved parts of the question well.

7.1.1 Object Detection

The overall object detection part with YOLO worked quite well in the research. The results in section 6.2 showed this with an average accuracy of 92,4% for detecting vehicles. However, it struggled more with the classification of the vehicles, which lead to the position in lanes to be less accurate. Almost 60% of the errors were caused by the detector misclassifying vehicles. As stated in section 5.1.1, the network was trained on the COCO data set [46]. COCO was a data set that contained 80 different objects, and not only objects related to traffic [79]. A network more specialized for detecting vehicles and traffic objects could possibly contribute to improving the classification accuracy, thus lowering the error rate and therefore boost the accuracy of the system.

As described in section 3.2.1, there were multiple different algorithms that could be used for object detection. YOLO was the best one as stated in section 3.2.2, but also other algorithms, such as RetinaNet [49], could possibly have worked better to increase the accuracy. YOLO was chosen because of its speed, so a possible drawback of choosing another algorithm would be longer processing time.

Another solution could be to just track parts of the vehicle, such as license plates [64]. License plates have standardized sizes which could make distance estimation more stable as well since the distance would be calculated from a measure that is equal for all vehicles. The drawback of this approach would be that the license plates become too small when the distance becomes too long. Using the license plate was also only tested on stationary cameras, and it was uncertain how accurate this would be for a moving camera. Using another reference to the detect the objects would, however, require both data collection, labeling of data, and retraining of the object detector. This would also remove the capability the detect different vehicle types.

Limitations

One limitation of the object detection could be the used of a pre-trained neural network. As stated earlier, the COCO data set was trained on a lot more objects than needed for this problem, and a more specialized network with only vehicles could have worked better.

Finally, the bounding boxes created around the object was sometimes unstable and varied more than desired between frames. The reason for this was YOLOs' way of detecting the bounding boxes and using non-max suppression to eliminate multiple bounding boxes for the same object [44].

7.1.2 Lane Detection

Another key feature of the system was the ability to detect lanes and separate vehicles in the different lanes. In our experience of testing the system, this worked well on straight roads when the lane markings were clear and easily visible. As the related work mentioned in section 3.3 about HT, the results confirmed that HT with Canny Edge detection worked as expected to detect the lanes. However, the overall lane detection did not work satisfiable on lane markings with much curvature as the output of the lane detection algorithm was exclusively straight lines. The ROI part that [60] proposed in their paper worked well and excluded many false lane detections since only the road was in focus.

The edge detection part with Canny Edge, proposed by several papers of related work [59, 60, 61], worked as expected. The edge detection technique managed to find the appropriate edges when the markings were good enough, but sometimes the edge detector also detected curbs as a lane edge.

The results and evaluation demonstrated that wrongfully detected lanes were the second highest source of error. Some of the problems with the lane detection would be difficult to improve, such as lanes far away being difficult to detect and distinguish, and bad or non-existing lane markings. Some improvements could, however, be done. It was not given that HT was the best choice when choosing the line detection part of the algorithm. The method proposed by Kim [53], of tracking left and right lane markings separately and utilizing an ANN that were trained to detect lines could possibly have increased the accuracy. This would also have the added benefit of being able to detect lanes with curves. However, using an ANN would potentially increase the resource demand and would also require training.

Another potential improvement could be to utilize standardized lane sizes in the algorithm. The Norwegian Public Roads Administration had a handbook [86] with different standard sizes of lanes and markings. This could ensure that different lane detections were not bigger than a set threshold.

Limitations

The lane that was detected on the left side was often the lane with opposing traffic, which was a problem that often created noise in the results. Also, when there was a central reservation that separated the two traffic directions the lane became unclear to the detection algorithm.

Another limitation of the lane detection was the conditions of the road. Since the edge detector needed clear markings, both worn out markings and weather conditions such as snow would potentially have an impact on the system performance. We were not able to test the system on snowy conditions, but the possibility of complications of detecting lanes would be high. Light conditions would probably also have an impact on the performance, but not as much as worn out markings and weather-related issues.

The lane detection had problems with separation of lanes near the vanishing point and when the lanes were curved. The system was also only able to separate vehicles in three lanes, by essentially finding the two lane lines for the lane the camera vehicle were situated in, and placing other vehicles in either side of these lines. For the system to work properly on bigger roads, the system should be able to detect and place vehicles in more than three lanes. This would require an expansion of the algorithm.

7.2 Combination of Height and Width when Calculating Distance

This section will discuss the distance estimation part of the system, which is tied to Research Question 2. The results were presented and evaluated in section 6.3.

To estimate distance in images with only 2D-information, we used the pinhole model, as described in section 3.4.1. The pinhole model required the true size value of the object to be known to calculate the distance. Vehicles in traffic had varying sizes, so to calculate the distance for different vehicle types, the system had to differentiate which values it used for the calculation based on the type of vehicle. The values used were listed in Table 5.1 from section 5.2.1.

The results demonstrated that a combination of 85% of the distance calculated based on the height, and 15% of the distance calculated based on the width, gave the lowest amount of error with 11% on average.

The same camera, with the same settings, that was used to record our traffic footage, was also used for this experiment. This ensured that the behavior of the footage for this experiment was equal to the rest of the video footage used in this thesis, which was important for the transferability of the result.

7.2.1 Limitations

When the system was to measure the distance of vehicles in traffic, the object detection part of the system caused some issues. Due to the camera movement and varying sizes of the generated bounding boxes, which could vary from frame to frame, the pixel sizes of the detected vehicles would also vary. This caused the actual distances estimates of vehicles in traffic to vary a lot more than when they were detected on stationary vehicles in the experiment.

However, since we did not have any advanced equipment to measure the true distance to vehicles while we drove and captured our test data, we had to rely on stationary vehicles to measure and calculate the best ratio. More accurate results could be achieved by conducting tests on moving traffic, but that would require the use of more advanced equipment such as a LIDAR, which as stated in section 3.4.3 are more accurate when calculating distances due to the available 3D information it generates.

Using a set of fixed values for the true vehicle sizes were also a limiting factor for distance estimation. As discussed earlier, these values were rarely 100% right and contributed to some error in the distance measure. The use of calculated true values would probably have given higher accuracy but would require stable reference points which could be challenging to locate and utilize.

Wrongly classified vehicles were a limitation as well. This was tied to the limitation of the object detector and was discussed previously. It was however, also a source of error for this part of the system. Wrongfully classified objects meant that the wrong, true measures were used to estimate distance. An improvement of the object detection part could also bring refinement to distance estimation.

7.3 Estimate Speed Using Distance

This section will discuss Research Question 3, which was about ”How can speed of the nearby vehicles be estimated by using distance calculated from image data captured by a vehicle-mounted monocular camera.” The results and evaluation belonging to this question were presented in section 6.4.

The speed estimation worked as desired for vehicles in front of, and going the same direction as the camera, with a mean difference of 2.09 m/s. As mentioned in the evaluation part, we did not use any extra equipment to measure the real speed of the target vehicle. Instead, we calculated a comparable speed based on GPS data and knowing the FPS of the videos. Incorporating GPS data should suffice, but because we also based the calculations on the distance estimation done by the system, it was plausible that the real average difference could be higher.

While using and analyzing the different parts of the system, we experienced that the speed estimation of vehicles other than those which moved the same direction as the camera, was not not as accurate. A probable reason was that the vehicles driving in the opposite direction were often in view only a short period, which gave fewer measurements for calculating the speed accurately.

Another inaccuracy we experienced was that, because the camera was in motion, stationary vehicles appeared to have movement. This caused them to be wrongfully assigned a speed other than zero. The reason for this was that even though the vehicles were stationary, e.g., waiting on a traffic signal, the bounding box would change in size as an effect of the movement of the camera.

The issue with the vanishing point and vehicles far away being too small to achieve accurate estimations also applied to the speed estimation.

Accurate speed estimation relied on accurate distance estimation. As stated in Section 7.2, these estimations were not always stable. The speed estimation would work as expected as long as the distance measure was consistent and stable. A significant error in estimated speed occurred if the estimated distance suddenly spiked or varied between consecutive frames. To combat the varying distance estimation, it was chosen to calculate an average to get the change in distance per time. This made the fluctuations much smoother and prevented the small inconsistency between consecutive frames having too great of an impact on the speed. The drawback of this was seemingly a lag in the speed estimate, where the system would take up to a couple of seconds before the speed was adjusted in the event of a sudden change of speed. It could be possible to adjust the size of the window used to calculate the average distance change, but reducing it too much could cause the speed to become too erratic.

7.3.1 Limitations

The research demonstrated that it was feasible to estimate the speed of nearby vehicles, using just image data from a vehicle-mounted camera, and the implementation itself answers how it could be done. However, as mentioned above, it had its limitations and challenges. Just the task of extracting 3D information from a 2D image was dependent on a lot of coherent factors. When the motion was to be incorporated into the equation as well, the problem became even more complicated.

As mentioned earlier, the delay in update of the estimated speed should be dealt with in case of future iterations, as it was a limitation for the system in its state then. If the distance estimation were less erratic, it would perhaps not necessary to use a running average, and the speed estimation would maybe be more responsive.

An issue we experienced with the system was that the tracking of already detected vehicles was sometimes unstable, especially in sharp turns. This led to assigned IDs of vehicles being lost and the vehicles reassigned new IDs. An unwanted effect of this was that the history was lost and the new IDs had fewer data to calculate averages from, causing greater fluctuations than desired. A more fine-tuned object detector could prevent this from happening by not losing the already detected vehicle longer than the system could manage without error.

7.4 General Discussion

This section will discuss the overall research without limiting the discussion to the research questions. Here topics such as the general system and privacy are taken into consideration.

7.4.1 System in General

A system was built using different algorithms and image processing techniques that answered the different research questions. The implementation and pipeline of this system were described earlier in Section 5.4. As the results from RQ1 demonstrated, it did not perform as well as we hoped concerning the vehicle counting ability. The error rate was higher than we expected, and the system needs some adjustments before working as well as this type of systems should.

The idea of the system was to the best of our knowledge a new way of using these algorithms for extracting traffic data since most of the Related Work (Chapter 3) were using stationary cameras or relied on other sensors for speed and distance estimation. We did our research with a vehicle-mounted camera, with the challenges that brought with it. Other research that was done with a moving vehicle [66, 68, 69], focused on a single vehicle in front and were not concerned with the general traffic situation, or required V2I-information.

The combination of these technologies is what made these results unique, and the system could be viewed as the first iteration of something that could be utilized in the future.

Focusing this thesis on a monocular camera was a deliberate choice since we wanted to investigate its possibilities, and discover eventual limitations. Utilizing a stereoscopic camera could possibly improve the performance of the distance detection part of the system due to the ability to use geometry for distance estimation. However, it could also complicate other aspects and further increase the demand for computing power required. Some challenges with using a stereoscopic camera would be camera calibration, image synchronization, and the need for a parallel object detection pipeline.

7.4.2 Computational Performance

The computational performance of the system must also be discussed. Even though it has not been a core part of the research, the system was developed with performance in mind. Using a state-of-the-art object detection algorithm as a foundation for the system, especially when it comes to computational speed, was a deliberate choice early on. Also, the programming language and frameworks were selected with performance in mind, since they allowed the usage of a GPU for increased computational power. Even though the system was not going to be used in safety critical applications, it was still essential to achieve acceptable operation times. Without going too much into details, the system ran at around 10-15FPS on a medium to high-end desktop computer, with an Intel i7-7700k CPU and NVIDIA GTX 1080ti GPU. This gave a processing time of 60-100 ms per frame. Considering the videos were captured at 30 FPS, this meant that when running the system, the performance was equal to the videos playing at half the speed.

To speed up the system and increase the performance, it could be possible to only process every other frame, effectively doubling the frame rate and achieving real-time calculation speeds. Based on some small tests conducted during development, the effect of this caused some measurements to become more unstable due to more significant differences between processed frames. To fix this, it would require some changes in the way some parts of the system operates, but it should be attainable. This is something that could be considered if the system were to undergo further development, and real-time performance was desired.

7.4.3 Privacy

Privacy was a concern when researching on topics where people could be identified, and sensitive data about them could be stored. Since we developed a system that used videos of traffic, we captured people driving and the license plates of many vehicles. It was also possible that situations where people were involved, could occur and that people were recorded unwillingly and unknowingly. To deal with this, The Norwegian Center for Research Data (NSD) [87] was contacted with an application of allowance to research image data with mentioned content.

The answer we received from NSD was that this research was not a concern as we did not develop a system that handled sensitive information.

To stay on the safe side, we chose to manually blur the license plates of vehicles used in this thesis anyway, and not use images where people could be recognized. An algorithm that automatically blurs the license plates could be considered if footage should be released in the future, even if the project is not required to have an NSD approval. The videos and data used were also only stored on our computers and systems, to not involve any external parties that may have been non-compliant with NSD.

Chapter 8

Conclusion and Future Work

This chapter presents the conclusion of each research question and takes the scientific contribution and future work into consideration.

8.1 Object and Lane Detection

The object and lane detection part of the system tried to answer the first research question, which was: ”How can the number and type of vehicles in nearby lanes of a vehicle be found using image data captured from a vehicle-mounted monocular camera?”.

The results from this question demonstrated that the results were not perfect, but the question was answered with a working proof of concept that was able to produce a general overview of the traffic surrounding the vehicle. There was to the best of our knowledge, no other research or application that utilized object detection coupled with lane detection, from a vehicle point of view, to achieve this.

Our contribution on this topic was to test the possibilities for a system that did all this combined.

Future work on this question was mainly about performance and how to lower the error rate in real traffic. The system will need an iteration to be able to work in a broader environment and consider more lanes. More specialization of the object detector is another point for improvement, and this could as described earlier, possibly be done with training on a data set with fewer classes. The lanes detected could also be approved by possibly using standard lane sizes to set a threshold of size and look at ways to detect curved lanes. Since our system was based on a single forward facing camera, it was only able to detect other vehicles in front of the vehicle. A possible extension of this system could be to additionally use a backward facing camera to get complete information about the surrounding traffic. In theory, the system does not need that many adjustments to be able to incorporate this.

8.2 Distance Estimation

Distance Estimation was an essential part of the system and was closely related to Research Question 2, which was: ”What is the most accurate combination of width and height when calculating distance using image data captured from a vehicle-mounted monocular camera.”.

The results and discussion show that a combination of 85% distance measured from using the height and 15% of the distance measured from the width of the vehicle worked best when calculating a measured distance.

Our contribution with this research question was this ratio to use when using the pinhole model to estimate distance in a traffic environment.

Future work would be to measure the ground-truth in traffic, and see if it was confirmable that this was the best ratio.

8.3 Speed Estimation

Speed Estimation was a part of Research Question 3, which was: ”How can speed of the nearby vehicles be estimated by using distance calculated from image data captured by a vehicle-mounted monocular camera.”

The results and discussion show that measuring the speed of vehicles driving the same direction as the filming vehicle was doable with just a small error margin. This results assumed the distance estimation to be correct.

Our contribution with this research question was proof of speed estimation being feasible from a moving vehicle using only a monocular camera.

To further improve this part of the system, the limitations discussed in Section 7.3 were issues that should be corrected to create a better system.

Also looking at other methods for the actual speed estimation could be of interest when dealing with unstable distances.

8.4 System in General

The system created to answer the research questions utilized state-of-the-art object detection algorithms and Computer Vision methods in an attempt to extract information about nearby traffic. The different components worked well together, and this thesis demonstrated that creating such a system was possible. The system had a lot of potential and managed to some degree to find the information required for a V2I system and intelligent intersection management. With future work, the system might be able to gather more of the information needed from Table 2.2.

Bibliography

[1] K. Bimbraw, “Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology,” in *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, vol. 01, July 2015, pp. 191–198.

[2] K. Peffers, T. Tuunanen, C. Gengler, M. Rossi, W. Hui, V. Virtanen, and J. Bragge, “The design science research process: A model for producing and presenting information systems research,” *Proceedings of First International Conference on Design Science Research in Information Systems and Technology DESRIST*, 02 2006.

[3] Council of European Union, “Council directive (EU) 2010/40/eu,” 2010. [Online]. Available: https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2010:207:0001:0013:EN:PDF

[4] L. A. Klein, M. K. Mills, and D. R. P. Gibson, Oct 2006, ch. Traffic detector handbook : third edition. Volume I, tech Report. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/954

[5] C. Chai and Y. D. Wong, “Automatic vehicle classification and tracking method for vehicle movements at signalized intersections,” in *2013 IEEE Intelligent Vehicles Symposium (IV)*, June 2013, pp. 624–629.

[6] H. Rabiu and H. Bashir, “Intelligent traffic light system for green traffic management,” 12 2015.

[7] O. Younis and N. Moayeri, “Employing cyber-physical systems: Dynamic traffic light control at road intersections,” *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2286–2296, Dec 2017.

[8] A. Guin, M. Hunter, M. Rodgers, J. Anderson, S. Susten, and K. Wiegand, “Integrating intersection traffic signal data into a traffic monitoring program,” *Transportation Research Record*, vol. 2593, no. 1, pp. 74–84, 2016. [Online]. Available: https://doi.org/10.3141/2593-08

[9] S. Lavrenz, J. Sturdevant, and D. Bullock, “Strategic methods for modernizing traffic signal maintenance management and quantifying the impact of maintenance activities,” *Journal of Infrastructure Systems*, vol. 23, no. 4, p. 05017004, 2017. [Online]. Available: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29IS.1943-555X.0000361

[10] M. Moshiri and J. Montufar, “Evaluation of detection sensitivity and count performance of advanced vehicle detection technologies at a signalized intersection,” *Journal of Intelligent Transportation Systems*, vol. 21, no. 1, pp. 52–62, 2017. [Online]. Available: https://doi.org/10.1080/15472450.2016.1198700

[11] P. G. Michalopoulos, “Vehicle detection video through image processing: the autoscope system,” *IEEE Transactions on Vehicular Technology*, vol. 40, no. 1, pp. 21–29, Feb 1991.

[12] J. A. Bonneson and M. M. Abbas, “Video detection for intersection and interchange control,” Sep 2002. [Online]. Available: https://tti.tamu.edu/tti-publication/video-detection-for-intersection-and-interchange-control/

[13] J. Reynolds, “Cruising into the future,” 2001. [Online]. Available: https://www.telegraph.co.uk/motoring/4750544/Cruising-into-the-future.html

[14] S. A. Bagloee, M. Tavana, M. Asadi, and T. Oliver, “Autonomous vehicles: challenges, opportunities, and future implications for transportation policies,” *Journal of Modern Transportation*, vol. 24, no. 4, pp. 284–303, 2016. [Online]. Available: https://doi.org/10.1007/s40534-016-0117-3

[15] M. E. Campbell, M. Egerstedt, J. P. How, and R. M. Murray, "Autonomous driving in urban environments: approaches, lessons and challenges." *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 368 1928, pp. 4649–72, 2010.

[16] S. International, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," 15/06/2018 2018. [Online]. Available: https://doi.org/10.4271/J3016_201806

[17] Volvo, "Forward thinking," 2017. [Online]. Available: https://www.volvocars.com/intl/discover-volvo/forward-thinking?module=true

[18] R. Browne, "Volvo and baidu join forces to mass produce self-driving electric cars in china," 01/11/2018 2018. [Online]. Available: https://www.cnbc.com/2018/11/01/volvo-baidu-to-mass-produce-self-driving-electric-cars-in-china.html

[19] "Autonomous cars to be in production by 2021," 2016. [Online]. Available: https://news.bmw.co.uk/article/autonomous-cars-to-be-in-production-by-2021/

[20] Ford, "Looking further - ford will have a fully autonomous vehicle in operation by 2021," 2016. [Online]. Available: https://corporate.ford.com/innovation/autonomous-2021.html

[21] ——, "Ford targets fully autonomous vehicle for ride sharing in 2021; invests in new tech companies, doubles silicon valley team," 2016. [Online]. Available: https://media.ford.com/content/fordmedia/fna/us/en/news/2016/08/16/ford-targets-fully-autonomous-vehicle-for-ride-sharing-in-2021.html

[22] The Tesla Team, "Dual motor model s and autopilot," 10/10/2014. [Online]. Available: https://www.tesla.com/blog/dual-motor-model-s-and-autopilot

[23] ——, "Your autopilot has arrived," 14/10/2015. [Online]. Available: https://www.tesla.com/blog/your-autopilot-has-arrived

[24] Tesla Motors, "Full self-driving hardware on all cars," 2018. [Online]. Available: https://www.tesla.com/autopilot

[25] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 289–299, Aug 2014.

[26] E. Uhlemann, "Introducing connected vehicles [connected vehicles]," *IEEE Vehicular Technology Magazine*, vol. 10, no. 1, pp. 23–31, March 2015.

[27] A. P. Chouhan and G. Banda, "Autonomous intersection management: A heuristic approach," *IEEE Access*, vol. 6, pp. 53 287–53 295, 2018.

[28] M. Khayatian, M. Mehrabian, and A. Shrivastava, "Rim: Robust intersection management for connected autonomous vehicles," in *2018 IEEE Real-Time Systems Symposium (RTSS)*, Dec 2018, pp. 35–44.

[29] C. Priemer and B. Friedrich, "A decentralized adaptive traffic signal control using v2i communication data," in *2009 12th International IEEE Conference on Intelligent Transportation Systems*, Oct 2009, pp. 1–6.

[30] D. Jiang and L. Delgrossi, "Ieee 802.11p: Towards an international standard for wireless access in vehicular environments," in *VTC Spring 2008 - IEEE Vehicular Technology Conference*, May 2008, pp. 2036–2040.

[31] S. Gräfling, P. Mähönen, and J. Riihijärvi, "Performance evaluation of ieee 1609 wave and ieee 802.11p for vehicular communications," in *2010 Second International Conference on Ubiquitous and Future Networks (ICUFN)*, June 2010, pp. 344–348.

[32] E. ETSI, "Draft 302 637-2 v1.4.0 intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service," *ETSI, Sept*, 2018.

[33] F. Alesiani, O. M. Lykkja, A. Festag, and R. Baldessari, "Cooperative its messages for green mobility: An overview from the ecomove project," 2012.

[34] C. Wuthishuwong and A. Traechtler, "Vehicle to infrastructure based safe trajectory planning for autonomous intersection management,"

in *2013 13th International Conference on ITS Telecommunications (ITST)*, Nov 2013, pp. 175–180.

[35] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[36] W. S. Sarle, "Neural networks and statistical models," 1994.

[37] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85 – 117, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608014002135

[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[39] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436 EP –, May 2015. [Online]. Available: https://doi.org/10.1038/nature14539

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. USA: Curran Associates Inc., 2012, pp. 1097–1105. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999134.2999257

[41] D. H. T. Hien, "The modern history of object recognition - infographic," Apr 2017. [Online]. Available: https://medium.com/@nikasa1889/the-modern-history-of-object-recognition-infographic-aea18517c318

[42] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun 2014. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2014.81

[43] M. Everingham and J. Winn, "The pascal visual object classes challenge 2012 (voc2012) development kit," *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep*, 2011.

[44] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[45] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," pp. 91–99, 2015. [Online]. Available: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks

[46] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[47] ——, "Yolo9000: Better, faster, stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[48] Y. Zhou, H. Nejati, T.-T. Do, N.-M. Cheung, and L. Cheah, "Image-based vehicle analysis using deep neural network: A systematic study," *2016 IEEE International Conference on Digital Signal Processing (DSP)*, pp. 276–280, Oct 2016.

[49] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," *arXiv*, Aug 2017. [Online]. Available: https://arxiv.org/abs/1708.02002

[50] A. Akoum, "Automatic traffic using image processing," *Journal of Software Engineering and Applications*, vol. 10, pp. 765–776, 01 2017.

[51] M. Pancharatnam and U. Sonnadara, "Vehicle counting and classification from a traffic scene," 09 2008. [Online]. Available: https://www.researchgate.net/publication/234136865_Vehicle_Counting_and_Classification_from_a_Traffic_Scene

[52] R. Uppala, "Vehicle detection and tracking using a support vector machine for autonomous vehicle applications," *Medium.com*, 08 2017. [Online]. Available: https://medium.com/@techreigns/vehicle-detection-and-tracking-using-a-support-vector-machine-for-autonomous-vehicle-applications-56a8d519fee8

[53] Z. Kim, “Robust lane detection and tracking in challenging scenarios,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 16–26, March 2008.

[54] S. Gupta, S. G. Mazumdar, and M. T. D. Student, “Sobel edge detection algorithm,” 2013.

[55] A. Seif, Mohammad Mohammadpour Salut, and M. N. Marsono, “A hardware architecture of prewitt edge detection,” in *2010 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology*, Nov 2010, pp. 99–101.

[56] L. Ding and A. Goshtasby, “On the canny edge detector,” *Pattern Recognition*, vol. 34, no. 3, pp. 721 – 725, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320300000236

[57] G. Shrivakshan and C. Chandrasekar, “A comparison of various edge detection techniques used in image processing,” *International Journal of Computer Science Issues*, vol. 9, pp. 269–276, 09 2012.

[58] J. Vijayakumar and L. J. Durai, “A review and performance analysis of image edge detection algorithms,” *International Journal on Future Revolution in Computer Science Communication Engineering*, vol. 3, no. 12, pp. 397 – 401, 2017. [Online]. Available: https://pdfs.semanticscholar.org/c59d/427b040a88cd8d358766e6130c701d6964c0.pdf

[59] Yue Dong, Jintao Xiong, Liangchao Li, and Jianyu Yang, “Robust lane detection and tracking for lane departure warning,” in *2012 International Conference on Computational Problem-Solving (ICCP)*, Oct 2012, pp. 461–464.

[60] G. Deng and Y. Wu, “Double lane line edge detection method based on constraint conditions hough transform,” in *2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, Oct 2018, pp. 107–110.

[61] L. Dang, G. Tewolde, X. Zhang, and J. Kwon, “Reduced resolution lane detection algorithm,” in *2017 IEEE AFRICON*, Sep. 2017, pp. 1459–1464.

[62] R. K. Satzoda, S. Sathyanarayana, T. Srikanthan, and S. Sathyanarayana, “Hierarchical additive hough transform for lane detection,” *IEEE Embedded Systems Letters*, vol. 2, no. 2, pp. 23–26, June 2010.

[63] J. Matas, C. Galambos, and J. Kittler, “Robust detection of lines using the progressive probabilistic hough transform,” *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 119 – 137, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314299908317

[64] W. Wu, V. Kozitsky, M. E. Hoover, R. Loce, and D. M. Todd Jackson, “Vehicle speed estimation using a monocular camera,” *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 9407, 03 2015.

[65] K.-Y. Park and S.-Y. Hwang, “Robust Range Estimation with a Monocular Camera for Vision-Based Forward Collision Warning System,” vol. 2014, p. 9, 2014. [Online]. Available: 10.1155/2014/923632

[66] J. Han, O. Heo, M. Park, S. Kee, and M. Sunwoo, “Vehicle distance estimation using a mono-camera for fcw/aeb systems,” *International Journal of Automotive Technology*, vol. 17, no. 3, pp. 483–491, Jun 2016. [Online]. Available: https://doi.org/10.1007/s12239-016-0050-9

[67] S. Russell and P. Norvig, “Artificial intelligence: A modern approach.” Upper Saddle River, NJ, USA: Prentice Hall Press, 2009, ch. 15.4.

[68] K.-Y. Park and S.-Y. Hwang, “Robust range estimation with a monocular camera for vision-based forward collision warning system,” *The Scientific World Journal*, vol. 2014, p. 1–9, 2014.

[69] A. Joglekar, D. Joshi, R. Khemani, S. Nair, and S. Sahare, “Depth Estimation Using Monocular Camera,” 2011. [Online]. Available: https://www.semanticscholar.org/paper/Depth-Estimation-Using-Monocular-Camera-Joglekar-Joshi/f3a8f252a3cd67ee61c28999b1021f1bed705b5b

[70] B. Schwarz, “Lidar: Mapping the world in 3d,” *Nature Photonics*, vol. 4, no. 7, p. 429, 2010.

[71] A. N. Catapang and M. Ramos, “Obstacle detection using a 2d lidar system for an autonomous vehicle,” in *2016 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, Nov 2016, pp. 441–445.

[72] L. Huang and M. Barth, “Tightly-coupled lidar and computer vision integration for vehicle detection,” in *2009 IEEE Intelligent Vehicles Symposium*, June 2009, pp. 604–609.

[73] J. Mrovlje and D. Vrančić, “Distance measuring based on stereoscopic pictures,” 2008.

[74] B. Lim, T. Woo, and H. Kim, “Integration of vehicle detection and distance estimation using stereo vision for real-time aeb system.” in *VEHITS*, 2017, pp. 211–216.

[75] K. Liu, Y. Ye, X. Li, and Y. Li, “A real-time method to estimate speed of object based on object detection and optical flow calculation,” *Journal of Physics: Conference Series*, vol. 1004, p. 012003, 04 2018.

[76] R. Sharma, “Measuring traffic speed with deep learning object detection,” *Medium.com*, 01 2019. [Online]. Available: https://medium.com/datadriveninvestor/measuring-traffic-speed-with-deep-learning-object-detection-efc0bb9a3c57

[77] L. Makarem and D. Gillet, “Information sharing among autonomous vehicles crossing an intersection,” in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2563–2567. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6378131

[78] “ayooshkathuria/pytorch-yolo-v3,” Jan 2019. [Online]. Available: https://github.com/ayooshkathuria/pytorch-yolo-v3

[79] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: common objects in context,”

CoRR, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[80] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” *CoRR*, vol. abs/1708.02002, 2017. [Online]. Available: http://arxiv.org/abs/1708.02002

[81] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” in *NIPS-W*, 2017.

[82] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[83] “Gopro hero 7 black.” [Online]. Available: https://shop.gopro.com/EMEA/cameras/hero7-black/CHDHX-701-master.html

[84] [Online]. Available: https://maps.google.com/

[85] “Gopro telemetry extractor.” [Online]. Available: https://tailorandwayne.com/gopro-telemetry-extractor/#!

[86] Vegdirektoratet, “Vegoppmerking,” 2015. [Online]. Available: https://www.vegvesen.no/_attachment/69741

[87] “NSD - Norwegian Centre for Research Data,” May 2019. [Online]. Available: https://nsd.no/nsd/english/index.html

Holthe-Berg, Lotsberg

Using Computer Vision to Extract Information About the Traffic Using a Vehicle-Mounted Monocular Camera

NTNU
Norwegian University of
Science and Technology