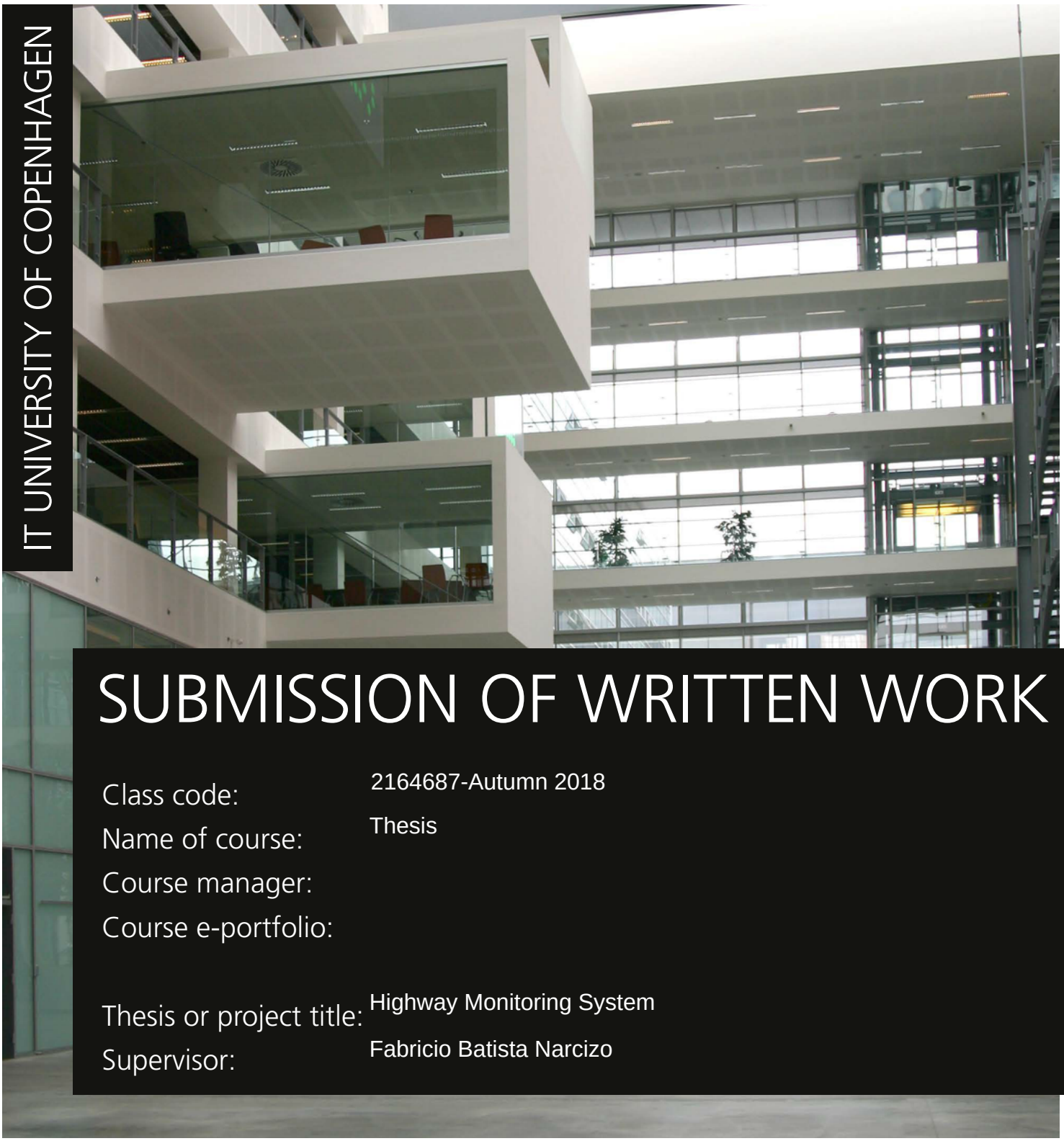
IT UNIVERSITY OF COPENHAGEN



SUBMISSION OF WRITTEN WORK

Class code:	2164687-Autumn 2018
Name of course:	Thesis
Course manager:	
Course e-portfolio:	
Thesis or project title:	Highway Monitoring System
Supervisor:	Fabricio Batista Narcizo

	Full Name:	Birthdate (dd/mm-yyyy):	E-mail:
1.	Andreas Jacobsen	15/12-1993	anjac@itu.dk
2.			@itu.dk
3.			@itu.dk
4.			@itu.dk
5.			@itu.dk
6.			@itu.dk
7.			@itu.dk
8.			@itu.dk

IT UNIVERSITY OF COPENHAGEN

Master Thesis

Highway Monitoring System

Author: Andreas Jacobsen
Supervisor: Fabricio Batista Narcizo

IT University of Copenhagen
Copenhagen, Denmark

A thesis submitted in fulfillment of the requirements for the degree of Master of Science.

January 2019

Abstract

The future of traffic calls for intelligent transport systems, which can handle the increased demands and expectations of future traffic. This thesis examines the possibilities for creating a highway monitoring system. Highway monitoring is important in the future because traffic will increase. The purpose of this thesis is to build and evaluate such a system with focus on high performance and accuracy while respecting the privacy of drivers on the highway. It presents the design of the system based on different image processing and machine learning methods, with evaluations of their individual performance and accuracy, as well as the overall performance of the system. The results show that it is possible to build a highway monitoring system on relative mundane hardware, while still attaining a specific accuracy above 90% in most cases and a generalized performance of above 100 FPS.

This thesis contributes to the ongoing research in intelligent transport systems by proposing a coherent highway monitoring system with great performance and strong privacy principles. It shows that privacy does not have to impact the functionality or performance of a highway monitoring system. It serves as an indication that privacy is a topic that is worth researching in combination with different systems. This thesis also discusses the implications of the proposed system for further investigation in validating and improving performance and in how different types of data collectors can work together to form a coherent highway monitoring system.

Keywords: Real-Time Monitoring, Highway Monitoring, Image Processing, Vehicle Recognition, Vehicle Tracking, License Plate Detection, Privacy.

Acknowledgments

First, I would like to thank my thesis supervisor Postdoctoral Researcher Fabricio Batista Narcizo of IT University of Copenhagen. He allowed me to make this project my own, while still guiding and supporting me. I would also like to thank him for his exceptional fast response time on e-mails, no matter when I sent them.

Finally, I am grateful to my parents and my girlfriend for giving me great support and encouragement through my years of studying and through this thesis.

Thank you.

Contents

List of Figures

List of Tables

1 Introduction

In modern societies, transportation infrastructure is essential for a variety of reasons. Many businesses rely on trucks to deliver supplies and employees rely on cars to get to work [1]. Also many private people rely on cars to get around [2]. The highways are central nerves of the infrastructure and support a large number of vehicles every day.

In Denmark, the number of vehicles on highways have grown over 31% since 2010 [2]. All those cars and trucks rely on the flow of traffic, which in turn relies on the ability of other drivers to follow the rules of the road, considering speed limits and other applicable rules. Sometimes, anomalies on the highways need to be handled by an appropriate office. This includes violation of speed limits as well as accidents blocking the road. Road users might also benefit from information about the current traffic flow on the highway to better plan their route.

Nowadays many of these things are handled by personnel manually monitoring the highway [3], either by watching the highway cameras or from a parked van in the side of the road, as done when controlling the speed of vehicles. It would be a benefit to automate this, preferable with existing equipment already present at highways. The detected anomalies would then be handed over to an appropriate office for investigation of the situation. Such a system would naturally need to include detection and recognition of vehicles, possibly using video feed from the highways in real time, to detect anomalies and report them. This is important in intelligent transportation systems, which is crucial for the future of transportation [4].

Wung, Chuang and Yi [5] propose a crowdsourcing-based system for real-time monitoring of highways. The system works by having a GPS and accelerometer installed in vehicles, in order to collect data about the roads. Other uses of accelerometer data include the architecture by Villanueva et al. [6], which supports a real-time monitoring system of the highway security fences, to detect a wide range of parameters. The wireless sensors are placed along the highway every 80-120 meters, measuring humidity, temperature, traffic, state of the perimeter fence, noise, among others. An alternative solution is proposed by Chen and Meng [7] using uncrewed aerial vehicles to detect vehicles on the road from aerial videos. This solution is more directed against specific deployments, such as if the police is searching for a car or for military operations, and not as a generic solution for daily tracking of highways.

All of these proposed systems require either specific hardware installed on the highway or in vehicles or require uncrewed aerial vehicles to fly over the roads. These examples are costly projects to deploy because of their specific requirements for hardware and placement. Alternative solutions include existing or easily deployed hardware, such as highway surveillance cameras. Such solutions depends on video images of the highway in order to detect and track vehicles. Since no hardware needs to be installed in the road, a camera based setup is also called a non-intrusive method. This makes it a preferable technology to use for gathering data, because highway cameras are easily deployed.

Currently, highway cameras are mostly manually observed 24 hours a day all year [3], which ends up being expensive. Most of this work can be automated, relying only on humans for reporting and investigation. A highway monitoring system should detect and track vehicles on its own, using machine learning and computer vision. The system should not replace humans, but it should make the job easier. Human involvement should still be required in all decision based on data from the system.

Highway monitoring systems consist of many different components all depending on different things. Common to all of the components is that they are all dependent on accurate detection of vehicles. Many researchers have worked with this topic, and the detection rate is often around 90% or higher [8][9][10]. Most of the current research studies compare how many vehicles the system counted in a video against how many vehicles drove by counted by a human. The system would logically also need to be able to track the vehicles and not just detect them, in order to determine factors such as speed.

Solutions have been proposed to solve the tracking problem, such as using the detected vehicle in the current frame as a template to search for it in the next frame [11]. Other options provide better possibilities for extensions, such as using the license plates as a unique ID for the vehicle [12]. The ID would allow the system to recognize it and also allow reporting to law enforcers about illegal actions or stolen vehicles. By doing object recognition it would also be possible to determine non-vehicle objects on the highway, which could increase the risk of accidents, such as pedestrians [13].

Another important aspect of highway monitoring systems is detection of a vehicle's speed. This is important for some reasons, such as determining violations of speed limits, but would also be important in many other aspects, such as traffic detection and accident detection. The first one requires an idea about the number of vehicles on the road but also their speed to determine the chance of a traffic jam. The second one could also use information about the speed of vehicles to detect an accident, together with the vehicle's direction of movement and a sudden change of it [14].

All of these components or features added to the same system would constitute a meaningful highway monitoring system. With the correct setup, such a system would be able to support the authorities in their day to day work with keeping the traffic flowing and the highways safe. As seen above many solutions exists for standalone problems, which are based on techniques using computer vision and machine learning.

1.1 Problem

There is little doubt that traffic will increase in the future and the need for intelligent transport systems will increase. Many researchers have proposed different solutions useful for such systems.

Looking at the studies discussed above, it is clear that there is a position for a coherent system for highway monitoring. Most studies only look at a standalone problem, such as vehicle recognition or license plate detection, and not the monitoring system as a whole. It is also relevant to look at a highway monitoring system, because of the manual work involved in traffic monitoring today. This calls for a more automated way of detecting and reporting traffic situations and could provide a greater granularity in traffic monitoring.

Another important observation is the lack of discussions about privacy and security in general for these systems. A system monitoring highway traffic would be a big privacy concern, and certainly an important tool for mass surveillance, and addressing it accordingly is essential.

This thesis proposes a coherent highway monitoring system, using existing technology, with strict privacy principles. The system should also be highly accurate, above 85%, and be able to run in real-time on hardware which can be placed on highways, such as cameras.

The goal of this thesis is to evaluate the current theories and methods relevant for a highway monitoring system, such as vehicle tracking and license plate detection. Furthermore it will investigate the methods for creating the system and analyze the system's performance and accuracy.

At last the thesis will also comment on hardware requirements for the system and on the privacy aspects of highway mass surveillance. The focus of the system is on how to prevent violating the general public's privacy, while still retaining a high accuracy.

1.1.1 Hypothesis

This study hypothesizes that a coherent highway monitoring system, which performs with an accuracy of at least 85%, can be build to improve traffic detection on highways using cheap existing camera hardware, while also focusing on protecting the privacy of the general public.

1.2 Approach

This thesis addresses the problem of creating a coherent highway monitoring system by first presenting a theoretical foundation for the understanding of monitoring systems on highways. The theoretical foundation focuses on methods for each functionality of the system: Vehicle recognition, vehicle tracking, license plate recognition, lane detection, speed detection, traffic detection and privacy.

This thesis proposes a working highway monitoring system build using existing technology. The development of the proposed system is done in Python, with support from image processing and machine learning libraries. The proposed system is build based on computer vision and machine learning methods described in the theoretical foundation.

The proposed system uses components and theory from previous studies, such as the ones discussed in this chapter, for its components. The evaluation of the proposed system depends on the accuracy of the machine learning algorithm and the performance of the overall system.

The proposed system is also build using strict privacy principles, which is in focus for every decision made in the design of the system.

The discussion of the proposed system is based on its performance and accuracy, the hardware requirements needed to run it and the privacy aspect of the proposed system and monitoring systems in general.

1.3 Outline

This thesis is divided into six chapters.

Chapter 1 introduces the thesis. The problem and hypothesis is presented and the foundation for the rest of the thesis is established.

Chapter 2 introduces the current research in highway monitoring and relevant theories for the proposed system. The chapter specifically looks at vehicle recognition methods and machine learning, vehicle tracking, license plate recognition, lane detection, speed detection, traffic detection, accident detection and privacy.

Chapter 3 goes through the development of the proposed system and the methods used. It presents three proposed systems: a video processing system, a highway monitoring system and a simulator. Each system is presented in details with their individual components and functionality. The chapter also proposes a traffic indicator used to describe traffic situations based on volume of vehicles and their average speed.

Chapter 4 presents the analysis of the proposed systems. The chapter evaluates the proposed video processing system and highway monitoring system. It validates performance and accuracy of the two systems, based on detection and recognition.

Chapter 5 discusses the proposed system. It discusses the shortcomings and challenges of the proposed system, the hardware requirements for the system to run, the privacy implications of monitoring systems, and what can be done to increase the security and privacy. The chapter also discusses future work for the proposed system.

Chapter 6 concludes the thesis and presents the outcome of the research. It answers the hypothesis proposed in this chapter.

2

Theory

Chapter 2 describes research previously done and evaluates its suitability for the highway monitoring system proposed in this thesis.

As described in Chapter 1 a highway monitoring system is a complex system composed of different components all playing an essential role in the performance of the overall system. Such components must be looked at both individually and as a single entity.

Section 2.1 discusses different methods for vehicle recognition and tracking. Vehicle recognition is perhaps the first and most important component of the whole system, seeing that all other functionality is dependent on this. Recognition of vehicles often depends on pre-processing of the video images.

Section 2.2 discusses license plate recognition. License plate recognition finds license plates from vehicles, which is important for identification.

Section 2.3 discusses lane detection. Lane detection makes it possible to determine lanes of vehicles and distances on the road using the road markings between lanes.

Section 2.4 discusses speed detection. Speed detection depends on the vehicle's movement over time and knowledge about distances on the road.

Section 2.5 discusses traffic detection. Traffic detection helps understand the number of vehicles on the road and the general speed of traffic.

Section 2.6 discusses accident detection. Accident detection finds traffic accidents using different methods.

Section 2.7 discusses privacy. Privacy is important because traffic systems can monitor drivers and be used for surveillance.

2.1 Vehicle Recognition

Recognition of vehicles in images and video is a broad topic with many methods and techniques. Features need to be extracted from the image in order to classify whether a vehicle is present or not. In the problem space, high-resolution images or video streams, are so large that it impacts the performance of the system.

Lately, Neural Networks (NN) are getting more and more traction, and many methods based on Convolutional Neural Networks (CNN) are getting popular. The benefit is their ability to detect all kinds of objects in images, not just vehicles, while still promising real-time performance.

Other options look at reducing the problem space, instead of searching the whole image or the entire video, as done with the convolutional neural networks. Object recognition is then done only on extracted areas for example around blobs extracted from the background or from a small area known to be on the road. Many of these approaches use machine learning to classify the extracted area as a vehicle or not [15].

Different approaches to extract moving objects from the background in images exist, such as temporal differencing and background subtracting. Temporal differencing is simple as it only takes two successive frames into account, but may result in poor extraction of features [16]. Another widely adopted approach, commonly used before trying to recognize vehicle, is subtracting an estimation of the background based on several frames in order to create the foreground mask [10][11][17][18].

Background Subtraction, also known as foreground detection, is the process of removing the background by keeping the objects which have moved from one frame to another. This is often done by building a background model of the scene and subtracting it with the current frame. Using multiple frames the algorithm estimates the background model. Several algorithms for background subtractions have been proposed, to improve the foreground mask and provide more robust results [19][20][21].

A suitable background subtraction algorithm should handle changes in the lighting of the scene, noise and keep a stable background model over a more extended period. As seen in previous research by Jacobsen and Bohr [22], algorithms based on a Mixture of Gaussian (MOG) show great promises, especially the MOG2 algorithm proposed by Zivkovic [20][21], which can also detects shadows of objects. Another algorithm getting traction because of resource constrained devices, such as the Raspberry Pi and embedded devices, is an algorithm based on counting proposed by Zeevi [23].

The general goal of background subtraction in vehicle recognition is to significantly reduce the problem space by only focusing on pixels that have changed from the last frame and thereby make it possible to distinguish moving vehicles on the road from the background road. Although, it is not sure that whatever is moving on the road is a vehicle. Therefore, further checks are required to ensure this. Another issue is that the foreground masks might not be easily understood as objects by itself and might require morphology to remove noise and blob extraction to distinguish the different pixels as blobs.

Binary Morphology is the process of manipulating binary images. The morphology depends on a structuring element, which is a simple binary shape that defines the operation on the binary image [24]. Often used structuring elements include different squares and crosses. Morphology is made up of two basic operations, erosion and dilation. Practically, this means making objects smaller and lines thinner with erosion or making objects larger and lines thicker with dilation. These two morphological operations can be combined for new operations, such as doing erosion followed by dilation, called opening, or doing dilation followed by erosion, called closing. The opening will open up gaps, separate objects and remove noise, while closing will fill small holes and enlarge regions.

Performing morphological operations on the binary foreground mask is a widely used technique to improve the performance of the final systems.

Blob Extraction is the process of detecting blobs in images based on regions with similar properties [25]. In vehicle recognition blob extraction is used to get properties from the foreground mask obtained using background subtraction. These blobs will represent moving objects, most likely vehicles on the road, and will provide useful properties for recognition. Soleh et al. [10] propose a solution using the blob properties directly as classification features, such as the blob contour and bounding box area. The research showed great promises in accuracy often above 80% of correctly detected vehicles, even during afternoon and rain [10]. This is considerably high compared with other methods, that only detected around 50% of vehicles in similar conditions [26][27]. All methods had a performance of at least 85% of correctly detected vehicles on a clear day. Previous research by Jacobsen and Bohr [22] has also shown that the bounding box can be used to extract a small region of the image with high performance for further classification using machine learning.

The methods described above are often part of the pre-processing step of most systems and does not provide the capabilities for full recognition of vehicles. Their purpose is to eliminate the need for machine learning algorithms to search the whole image and therefore simplify the problem space. In order to recognize whether an object is a vehicle or not, the system will use the method described above to clean and extract the relevant objects from the image and prepare them for further work, such as using the bounding box obtained from blob extraction in a following step. This next step will often be feature extraction and machine learning. Even systems relying only on machine learning to recognize vehicles in the whole image, such as systems using a convolutional neural network, will still require some pre-processing often to reduce noise in the image.

The next section discuss the extraction of features for the algorithms to learn and machine learning itself. The following sections also discusses tracking vehicles and recognizing license plates.

2.1.1 Feature Extraction

Feature extraction is the reduction of raw data into manageable data [28].

Before a system can learn to classify objects, it needs features to learn. The various methods can extract many different types of features from image and perform differently depending on the features extracted.

Harris Corner Detection detects a local neighborhood with two dominant edge directions [29]. Du et al. [17] proposes a solution that uses the corners obtained from Harris corner detection to classify whether a vehicle is a car, truck or bus. This is done by using three standard examples and calculating the Hausdorff distance from a newly seen vehicle to the three possible classification and letting the shortest distance determine the type of vehicle as either a car, truck or bus [17].

Histogram of Oriented Gradients (HOG), as seen in previous research by Jacobsen and Bohr [15], is a widely used feature extraction method proposed for vehicle recognition by many researchers. HOG is a feature descriptor, often used in object detection and works well for the detection of vehicles [22]. It works by counting the number of gradient orientations in different neighborhoods of an image. Although HOG is not tied to a specific machine learning algorithm, it is often proposed together with a support vector machine, as proposed by Dala and Triggs [30] in their paper on human detection. Using HOG the system can extract features of vehicles and learn to classify new objects seen as either a vehicle or not, using machine learning. Figure 2.1 shows an example of extracted HOG features from a car.

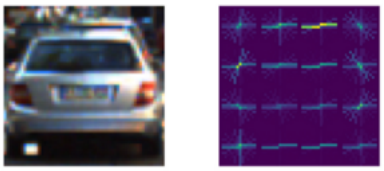


Figure 2.1: Example of HOG features extracted from a car [31].

2.1.2 Supervised Machine Learning

Machine learning allows machines to improve their performance on specific tasks [32]. An essential part is supervised machine learning, which allow the system to learn the features related to an object and classify it as belonging to a class, such as an object being a vehicle or not a vehicle.

Support Vector Machine (SVM) seems to be one of the most widely used machine learning methods for vehicle recognition, as seen in previous research by Jacobsen and Bohr [15]. It seems to perform at acceptable accuracy in many cases, providing an excellent option for many projects [22].

A support vector machine is a supervised learning model trained on a dataset of examples classified as either one class or another. This could be a dataset consisting of vehicles and non-vehicles classified manually.

SVM works by learning the parameters of a hyperplane separating the training data with the maximum margin to nearby data points. Vapnik and Chervonenkis [33] created the algorithm in 1963 as a linear model for classification, but in 1992, Boser et al. [34] suggested a method for nonlinear classification using SVM by applying the kernel trick. The kernel trick works by transforming the feature space to a higher dimension, allowing the hyperplane to be linear in the transformed feature space, while it may not be in the original input space [34].

Neural Networks in computer science models biological neural networks and are getting more and more popular [35]. They consist of neurons, which are trained to perform a specific task. Neural networks try to minimize a cost function until an optimal solution is found. This is done by changing the weights between neurons and recalculating the cost. This is how the network learns the optimal solution to a specific problem. Shi et al. [36] propose using neural networks for object detection from traffic cameras.

Previous research by Jacobsen and Bohr [22] has shown that neural networks, more specifically the multi-layer perceptron, holds great promises for vehicle recognition both with high accuracy and excellent prediction speed.

Method	Dataset	Training	Test	Prediction Speed
Decision Tree	0.978	1.000	0.925	00.01 ms/image
KNN	0.982	0.985	0.975	11.50 ms/image
Linear SVM	0.992	1.000	0.972	00.01 ms/image
MLP	0.992	0.999	0.975	00.05 ms/image
SVM	0.995	0.999	0.984	02.55 ms/image

Table 2.1: Comparison of different machine learning methods [22].

The prediction speed of the multi-layer perceptron has shown to far outperform the prediction speed of the support vector machine, by a factor of 50, while the accuracy is roughly the same (see Table 2.1).

Others, such as Maciejewski et al. [37], also propose using neural networks for vehicle recognition. In their paper, they compare the usage of a probabilistic neural network and a multi-layer perceptron, in the context of vehicle recognition.

Convolutional Neural Networks are proposed by researchers as a solution to solve real-time object detection. The groundwork was laid in 2014 by Girshick et al. [38] with the proposal of Region-based Convolutional Neural Networks (R-CNN), a method that combined region proposals and CNN. Girshick [39] later improved the method with a considerable increase in test-time performance, from the previous 47 seconds per image to less than 2 seconds per image. Later, Ren et al. [40] further improved on R-CNN to increase the test-time performance even more, to less than 1 second per image, and called the new method Faster R-CNN. He and Zeng [13] propose a real-time pedestrian warning system on highways using an R-CNN method.

Today, the more popular option is called YOLO [41][42][43], which stands for "You Only Look Once", claiming real-time performance for object detection on high-end graphics processors [44].

The downside of using these CNN-based methods is that they still require powerful hardware such as high-end graphics processors to run in real-time, although their usage is beneficial.

2.1.3 Vehicle Tracking

The process of identifying a previously known object based on its possible movement is called object tracking and lets the system predict the new position of an object instead of doing object detection on the whole frame again.

Template Matching is a technique to find a template image in another image. Unno et al. [11] propose template matching as a method for tracking a vehicle over several consecutive frames from a video. This is done by extracting the moving object area as a template and applying the template matching method to the next frame in the video.

Optical Flow describes the motion of objects between two consecutive frames, which makes it suitable to estimate a rough position for the vehicle in a given frame. Unno et al. [11] propose optical flow as a method to apply before template matching to avoid matching errors.

The use of optical flow generally falls into two categories, either it tracks a single point, or it tracks all points in the frame. The first case can be solved using the Lucas-Kanade method, which assumes that points in a local neighborhood have the same motion [45]. The result is an optical flow vector for points in the local neighborhood. The second case can be solved using dense optical flow, which computes the optical flow for all points in the video frame [46].

Kalman Filter is a popular method proposed by multiple researchers [18][47], as a robust tracking method to predict a vehicle's position in the next frame [48]. Baek et al. [47] use a Kalman filter but also extend on the method by using mean-shift tracking to find the best candidate and update the Kalman filter with the predicted vehicle position. Using vehicle tracking and vehicle recognition using a cascade classifier, Baek et al. [47] obtained a system performance of 48 frames per second on low-end hardware, a 1GHz processor with 1GB of memory, but in their work they encountered many false positives and planned to extend their solution with road surface detection to minimize such errors in the future.

2.2 License Plate Recognition

Recognition of a vehicle tells whether a detected object is a vehicle or not, but will not tell which specific vehicle this is. For solving that problem license plate recognition is an option since it is a well-known identification system for vehicles used worldwide. It also provides the option for finding stolen cars, as long as the license plate is still readable.

License plate recognition is the process of locating the license plate and identifying the numbers and can generally be divided into some algorithms, namely: Localization of the license plate, orientation and sizing of the license plate, normalization, character segmentation, and Optical Character Recognition (OCR) of the segmented characters [49].

Localization of the license plate can be a time-consuming algorithm depending on how the vehicle is recognized beforehand. Many researchers have different approaches for locating the license plate. Agui et al. [50] propose detecting the edges of a license plate using a Hough transform. Miyata and Oka [12] propose using a support vector machine for recognition whether something is a license plate or not. Other possibilities include using neural networks or template matching.

Character Segmentation requires the use of image processing to separate the characters on the license plate from the background. Segmentation of the characters can be done using adaptive thresholding with morphology and blob extraction. Miyata and Oka [12] discuss other options such as using grayscale template matching. Goel and Dabas [51] propose a solution based on blob extraction from a binarization of the image to obtain the bounding regions, with a reasonable accuracy even though they experiences difficulties for the system to decide between certain types of letters, such as D and O or Q and O. Goel and Dabas [51] also discuss options such as using a Hough transform to obtain the boundary lines and follow up with binarization techniques. Karthikeyan et al. [52] discuss many more options and present a comparison of the different algorithms performance, showing at least 94% of

accuracy for all algorithms. Character recognition could also be done without segmentation by using pattern matching directly on the located license plate [53].

Optical Character Recognition is the last step of license plate recognition and transforms the segmented characters into readable text for the system. In recognition of the characters, Goel and Dabas [51] proposes a method where the segmented characters are compared individually to character templates stored in a database. Likewise, Miyata and Oka [12] also discusses the option of using template matching on the binary image of segmented characters.

There also exists an open source library specialized in license plate recognition called OpenALPR [54], written in C++ and based on the Tesseract OCR engine [55] and OpenCV. It works by using pattern matching to locate the license plate and then perform OCR on the located region.

Using license plate recognition might not always be sufficient by itself and Tungkastan et al. [56] explain the benefits of extending the license plate recognition system with a vehicle model recognition system. Llorca et al. [57] propose HOG, together with support vector machine, as a method for recognizing vehicle logos from traffic images. This provides a better certainty for detection of suspicious vehicles and can also make it possible to compare the detected vehicle's make or model to the expected values as based on the license plate registration. A mismatch between registration and reality might lead to a stolen vehicle or other suspicious activity.

As seen above much research have been done in the area of vehicle recognition, allowing new systems to be build on good foundations, both in preprocessing, feature extraction, machine learning and uniquely identifying vehicles using license plate.

2.3 Lane Detection

Lane detection is the process of finding the straight lines surrounding lanes on roads. It is useful for many things, such as determining lanes for vehicles with lane departure warning systems or in self-driving cars. It is also relevant for highway cameras in order to determine the lanes vehicles drive in. Lane detection is built on the same principles whether the lanes are detected from a front-facing dashboard camera in a vehicle or from an overview of the road by a highway camera.

Many researchers propose Hough transform for finding the straight lines of lanes [58][59]. In order for Hough transform to detect the straight lines, the lines must first be extracted from the image. Two methods for processing the image before Hough transform are generally used: Thresholding or edge detection, such as sobel operator or canny edge detection. Low et al. [59] proposes using canny edge detection before a Hough transform. Canny edge detection works by finding the edges of the road lines, which acts as the points that can be transformed to lines by the Hough transform.

The other often used method, thresholding, works by setting a threshold for the image and thereby separating the colors between road and road markings, leaving only the markings. Straight lines can then be detected on the extracted pixels using Hough transform.

Researchers also propose methods for detecting curved lines [58][60], which is useful for curving roads. The highway camera might not always be placed at a straight road, even though highways generally curve less than other types of roads. Jang et al. [60] propose a method using the straight lines detected from Hough transform. Circles are generated on the road around the straight lines and the best candidate is found using voting based on the feature image extracted using pixel intensity. The curve of the best candidate circle represents the curved line of the lane.

The detected lines are used to construct the lanes, which are formed by the polygons between each set of lines.

2.4 Speed Detection

Vehicles are in motion on the highway, and the exact speed of their motion is relevant for many things, such as determining speeding or traffic jams. This section presents research done in vehicle speed detection and how it affects the overall system's functionality and performance.

Today, the standard way to determine speed on highways is to use radar system, which depends on specific hardware. Speed detection using image processing removes the high hardware dependency.

Gerát et al. [61] propose a system based on background subtraction for extracting the moving vehicles. The bounding boxes of the vehicles are then tracked with a Kalman filter and optical flow based on the Lucas-Kanade method, which allows the average velocity to be estimated. The actual speed is estimated using a method proposed by Alavianmehr [62], which determines the speed by dividing the real width of the road with its pixel representation, allowing the speed of pixels per second to be converted into kilometers per hour. The research done by Gerát et al. [61] shows that combining Kalman filter and optical flow, using the Lucas-Kanade method, provides a higher accuracy on detection of speed, compared with the two methods as standalone methods.

Other options include using corner detection, as proposed by Kumar et al. [63]. The method relies on background subtraction to extract the moving vehicles and then canny edge detection to find edges. The actual speed is found using corner detection, such that the distance between the maximum corners detected is calculated between two frames, allowing the system to convert the pixel distance per frame to actual speed [63].

Speed can also be detected using the road markings. With a known marking length, they can be used to estimate the distance moved for vehicles, which can be used to determine speed. On a highway with multiple cameras, the recognition of a vehicle can be used to determine the time it took to move from the first camera to the second.

2.5 Traffic Detection

Detecting traffic on highways is essential for providing traffic information updates to commuters or businesses who rely on the highway infrastructure. Traffic detection includes information about the general traffic flow, the density of traffic on the highway and traffic jams. It may also include the detection of traffic violations.

Pan et al. [64] propose a system for vehicle flow detection, using traffic detection based on background subtraction and edge detection. The road is then split into different lanes using lane detection, before doing vehicle counting on each lane based on the detected vehicles, to determine the traffic flow on specific lanes or the road in general [64]. Their research presents at least 90% accuracy under different kinds of illumination, even night [64]. Other researchers, such as Tahmid and Hossain [65] are also using edge detection to detect vehicles for determining the density of traffic on different lanes. They propose using a canny edge detector and comparing the detected edges from real-time traffic against the edges from an empty reference image to determine the density of the current traffic [65].

Lei et al. [66] propose a real-time traffic jam detection system using the speed of vehicles to detect traffic jams, done using background subtraction and Sobel filter edge detection to find moving vehicles. The general traffic flow speed is determined based on the transformation of the corners over time, and a traffic jam is detected once the flow speed gets below a defined threshold.

Traffic detection based on image processing mostly depends on vehicle recognition methods, such as background subtraction, edge detection, and speed detection. With the detection of vehicles and their speed, it is possible to determine the general traffic flow. Adding lane detection can provide the functionality to detect traffic jams or other problems in specific lanes impacting the flow of traffic.

2.6 Accident Detection

Accidents can greatly impact traffic on highways, and it is vital that such accidents get handled fast, especially when medical assistance is needed.

Accidents on highways often happen because of traffic violations, such as speeding or unsafe lane changes. Detection of speeding will depend on a speed detection system discussed in Section 2.4, while lane detection and vehicle recognition are necessary for detection of unsafe lane changes. A traffic violation detection system is proposed by Wang et al. [67] that can detect speeding, vehicles running red lights, the crossing of lanes and lines and vehicles moving backwards using background subtraction, edge detection and vehicle tracking.

Hui et al. [14] propose a traffic accident detection system tracking the movement of vehicles, based on the idea that the position, acceleration, and direction of movement of a vehicle will change rapidly over a short time when involved in an accident. The system depends on background subtraction, vehicle recognition, vehicle tracking and extraction of the essential parameters: position, acceleration, and direction of movement [14].

Walking pedestrians on the highway, is an issue which should be addressed immediately to avoid fatal accidents. He and Zeng (2017) propose a real-time pedestrian detection system.

The detection of accidents is at large dependent on the coverage of highway cameras and therefore shouldn't be the only system in place to handle traffic accidents. Accidents can happen between two highway cameras, and even though the accident might influence the traffic flow detected on the next camera, it might not be enough to address the issue fast enough. Although, it may give the law enforcement or healthcare an advantage in quicker locating the accident from a call by any involved in or witnessing the accident.

2.7 Privacy

Mass surveillance using monitoring systems, such as a highway monitoring system, can have large and widespread effects on the privacy of the general public. It is therefore important that such systems adhere to strict privacy and security principles.

Xie et al. [68] propose a solution where vehicles cannot be identified because their ID numbers are anonymized. No identifiable numbers are exposed to end users and used ID numbers are periodically refreshed. Others, such as Shigetomi et al. [69], propose using cryptographic anonymity to protect privacy and security of the system.

2.7.1 Privacy by Design

An important approach to system engineering is privacy by design, which makes privacy a focus through the whole design and development of a system.

Some of the important principles of privacy by design include: Respect for user privacy, privacy embedded into the design - it is not an add-on, proactive protection - not reactive and end-to-end security [70].

These principles should be taking into account when designing a highway monitoring system and ensuring that the drivers privacy are protected from mass surveillance. But even though privacy is central for the system it should not impact the general performance and functionality of the system.

Kung et al. [71] present some challenges relevant for protecting privacy in intelligent transport systems. First, systems might already be in place, which can hinder the adoption of a newer, more privacy friendly system. Second, a system needs to communicate with many different other systems, and those systems need to protect privacy as well. Each individual system must adhere to the same strict privacy principles. Lastly, not all stakeholders of the system might agree on a privacy friendly direction or agree on the use of common privacy principles across all components of the system.

3
Method

Chapter 3 describes the steps completed in order to build the proposed video processing system and highway monitoring system. The chapter goes through the stages of the two systems step by step to describe the methods for each of them. The systems are inspired by current theory and builds on previous research to avoid known pitfalls (see Chapter 2).

Section 3.1 describes the proposed video processing system. The purpose of the video processing system is to process frames from a video using image processing to extract the data needed for the highway monitoring system. The system is heavily dependent on image processing for extracting features and machine learning to classify them.

The section separates and describes the stages of the video processing in subsections: Background subtraction, lane detection, blob detection, vehicle recognition, speed detection, and license plate recognition.

Section 3.2 describes the proposed highway monitoring system. The primary purpose of the system is to monitor the traffic on highways and use the data gathered from the video processing system to predict the current traffic situation.

The section also proposes a new traffic indicator, which describes traffic situations based on volume of vehicles and their average speed.

Section 3.3 describes a simulator, which can simulate gathered data as an alternative to the video processing system for easier testing and evaluation. The simulated data is similar to the real-world data gathered from the video processing system but can be used to simulate different traffic situations.

3.1 Video Processing System

This section describes the proposed video processing system. The system is built using Python 3.6.6 with the following libraries: OpenCV 3.4.3, NumPy 1.14.6, Scikit-Learn 0.20.1. The system is built cross-platform.

The video processing system is started by a small program that consists of a file loader, a video player and a counter for frames per second. The small program loads a video file from the hard disk drive, passes it frame by frame to the video processing system, and plays the resulting video while measuring the frames per second. The measurements are useful for evaluating the video processing system's performance.

When the video processing system receives a frame from the video, it processes the frame in six stages. The stages result in detected and recognized vehicles with a timestamp, detected lane, detected speed, and detected license plate, which can be sent to the highway monitoring system for traffic detection (see Section 3.2). Figure 3.1 shows an overview of the six stages in the video processing system.

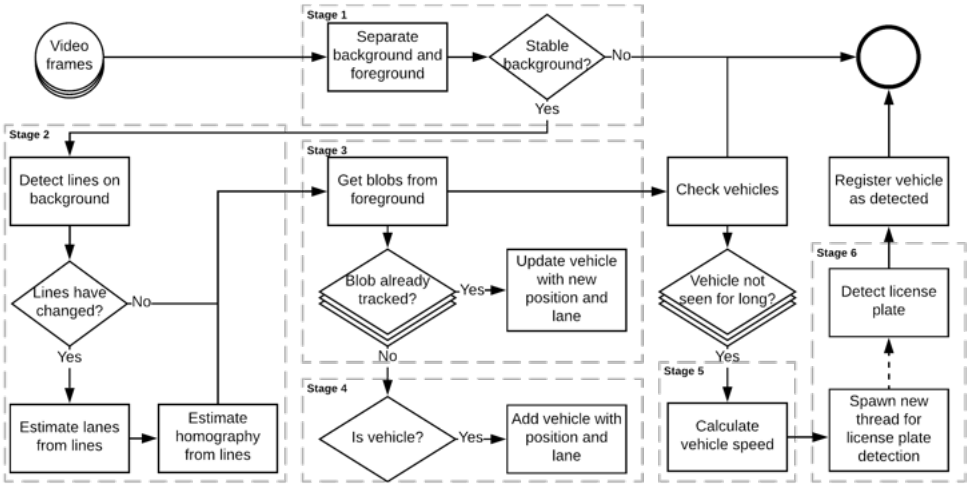


Figure 3.1: Overview of the video processing system.

The following sections are based on each of the six stages: Background subtraction, lane detection, blob extraction, vehicle recognition, speed detection, and license plate recognition.

Stage 1 separates the background from the foreground using background subtraction. In this stage, the system waits for a stabilized background before continuing. This ensures that the background and foreground contains less noise, which improves the results of the following stages. Section 3.1.1 describes the background subtraction in Stage 1.

Stage 2 finds the straight lines on the road. The lines are used to detect road lanes and estimates a homography used later for speed detection. After detecting road lanes and estimating the homography the first time, the system checks if the lines have changed from last frame before doing it again. This is to avoid recalculating every frame, while also providing support for moving cameras. Section 3.1.2 describes the lane detection in Stage 2.

Stage 3 uses the extracted foreground to detect blobs. Blobs are used for tracking vehicles based on their position in previous frames. The vehicle is updated with the blob's position when tracked. If a blob is not already tracked as a vehicle, it is first classified in Stage 4. Section 3.1.3 describes the blob detection in Stage 3.

Stage 4 classifies whether a blob is a vehicle or not, using supervised machine learning. This is done with every blob before considering it a vehicle. Section 3.1.4 describes the vehicle recognition Stage 4.

Stage 5 calculates the speed of vehicles. A tracked vehicle reaches Stage 5 if it has not been seen for an extended period and therefore considered to be out of the view. Speed detection is done using the estimated homography from Stage 2. Section 3.1.5 describes the speed detection in Stage 5.

Stage 6 recognizes license plates. This stage is done on a separate thread to improve performance of the system. When a license plate is recognized the vehicle is registered and can be sent to the highway monitoring system. Section 3.1.6 describes the license plate recognition in Stage 6.

3.1.1 Background Subtraction

In Stage 1, a background model is estimated using video frames in order to obtain moving objects from the video feed. The background subtraction is done using a background subtraction method focused on speed and based on Counting (CNT). It is available under the name `BackgroundSubtractorCNT` in the OpenCV library `bgsegm`. The background subtractor is using minimum pixel stability of 50 and maximum pixel stability of 3000 to determine when a pixel is part of the background and when it is part of a moving foreground object. The background subtractor can also be parallelized for added performance on multi-threaded systems.

The result of the background subtraction depends on morphology. Figure 3.2 shows the same scene with different results, based on whether morphology is used on the extracted foreground and which kernel size is used.

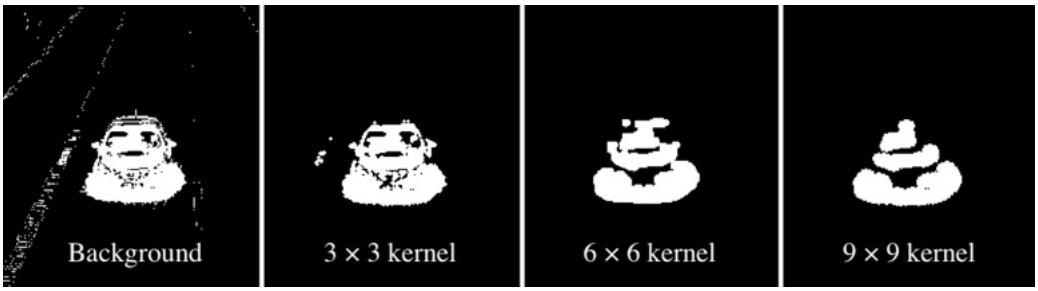


Figure 3.2: Background with morphology using different kernel sizes.

Morphology is used to reduce noise and enlarge objects extracted from the background for better detection. It is essential for the system as the foreground is filled with noise from other moving objects such as trees and birds. Because the camera might not stand completely still, this also generates noise in the images. The noise is removed using a morphological opening, that retains the size of larger blobs while removing smaller ones. A kernel size of 3×3 is used to provide the best result (see Figure 3.2).

3.1.2 Lane Detection

In order to provide the highway monitoring system with information about which lane a vehicle drove in, it is necessary to detect and extract the lanes from the road.

In Stage 2, the system automatically detects lanes by extracting the road markings from the background image. A simple thresholding method is used to extract the road markings, which are white on the gray road. The thresholding method removes all pixels with a value below 180 on the grayscale background image, which means it preserves all pixels making up the white road markings.

The extracted road markings can then be used to find the lanes by transforming them into lines. This is done using a Hough transformation which converts all the points into the most likely straight line through them. This method gives several different lines because of the nature of the road markings and noise. Many of the lines overlap or are only a small distance from each other, which makes it more difficult to discern the lane separations.

All the lines are reduced to only the necessary lines, for example three lines on a two-lane highway. The system do so by comparing the position of the lines. A line is removed if it is very close to another line. This ensures that the system only keeps lines which are far from each other. Figure 3.3 shows the detected lines, which form the sides of the highway lanes.

Figure 3.3: Example of detected lines (left) on a road image (right).

Next, the lines can be used to get the area of the lanes. By sorting the lines based on position, it is possible to form a rectangle using the two anchor points of the line and the two anchor points of the next line. Going through the lines then forms the polygons bounding the lanes, such that three lines give two polygons and four lines give three polygons. To estimate lanes correctly, the lines must be sorted in the right order or the lanes will be mixed up. This is done by calculating the angle of lines relative to the x-axis and sorting the lines by the dimension they are most spread out on.

The system recalculates the lanes if the detected lines change significantly, which is determined by the change of their angles. This is to support the possibility of moving cameras or other factors which might change the view of the lanes and ensures that the lanes are always correct.

Once the system recognizes a vehicle, it determines the lane it drives in. Detecting whether a vehicle is within one lane or another requires performing a check for whether the vehicle's position is within a specific polygon. If a vehicle cannot be determined to be within one lane or another it could be an error, because the vehicle is not detected correctly, because it has not entered the lane yet, or because the vehicle is driving in the emergency lane.

Figure 3.4 show the result of detecting the road markings and estimating the lanes on a two-lane highway.

Figure 3.4: Example of lanes detected on a two-lane highway.

3.1.3 Blob Detection

Once the system obtain a stable background using the previous background subtraction method in Stage 1, blob detection is applied to the foreground mask in order to obtain blobs from the video frame.

In Stage 3, blob detection finds blobs by extracting the contours of objects in the video frame. The blobs are used to track vehicles across frames.

The system uses contours to extract the size and shape of blobs, which is relevant for filtering out too small or too large blobs. The system also uses the bounding rectangle of the blob to extract the region of the original frame for further analysis, such as license plate detection in Stage 6. Figure 3.5 shows the bounding rectangle around a detected vehicle for visualization.

The detected blobs are compared with detected vehicles from the previous frame. If a blob is close to a vehicle's position in the last frame, it is considered the same vehicle. Using this method the system tracks vehicles across frames using the position of detected blobs.

If a detected blob is not close to a previously detected vehicle it is considered a new blob. A new blob is classified as either a vehicle or not a vehicle in Stage 4. If the blob is classified as a vehicle it is used in the following frames for tracking the vehicle. Every blob will be classified once to determine if it is a vehicle and tracked based on the blob position in consecutive frames.

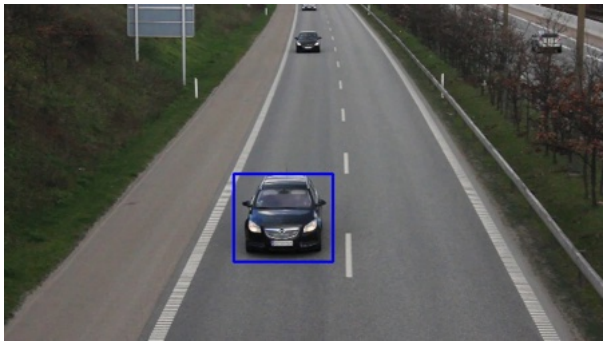


Figure 3.5: Example of a vehicle detected using blob extraction.

3.1.4 Vehicle Recognition

Once the blobs from the foreground mask have been detected, the bounding rectangle of a blob represents the area of a possible vehicle. The system does not initially know if an extracted blob is a vehicle or not and therefore have to perform additional checks to ensure this.

In Stage 4, a blob's bounding rectangle, the area of interest, is extracted from the video frame and processed by a supervised machine learning to determine whether the area includes a vehicle or not. This is only done on the extracted area of the blob and not the whole frame to speed up the performance of the system.

The classification depends on features extracted from the area of interest. The system extracts HOG features on a 64×64 cropped version of the frame only including the area from the blob extraction.

The vehicle is only recognized the first time it is seen and thereafter it is tracked based on the position of the blob, estimating if the new blob is the same vehicle as a previously recognized vehicle. This ensures that the time-consuming operation of predicting if a blob is a vehicle or not is limited to only the first frame of the vehicle and not all following frames.

This approach gives the system a better performance than checking the whole frame and recognizing every vehicle in all frames.

3.1.4.1 Training

The machine learning model is trained on a dataset consisting of 464 images of vehicles and 2790 non-vehicle images [72]. The data is learned by extracting HOG features from the vehicle and non-vehicle images and the features are then fed into a multi-layer perceptron with 3 layers of 13 neurons each. The dataset is split in two parts, such that the model is trained on 60% of the original dataset and the rest of the data is used for testing the accuracy in order to provide insight into the models performance. The training process is based on findings from previous research by Jacobsen and Bohr [22].

3.1.5 Local Speed Detection

In Stage 5, the video processing system has capabilities for local single-camera speed detection. This depends on detected lanes in order to determine the measurements of the road, and coupled with the tracking of vehicles on the road give estimations on vehicle speed. The precision of the blob extractions, which gives the tracking points for the vehicle speed, is essential for the precision of the detected speed. The speed is estimated based on the distance a vehicle moved on the road over a number of frames, converted to time, calculated from the frames per second. The distance is calculated from pixels on a measurable top-down view in correct scale constructed using the homography between the top-view and the road perspective in the video frames. Figure 3.6 shows the relation between the two views.

3.1.5.1 Homography

The homography between the real-world scene and the correctly scaled and warped top-down view is estimated from a series of steps. These steps go from the detected lines from the Hough transform to constructing intersection points across the road to estimating the homography.

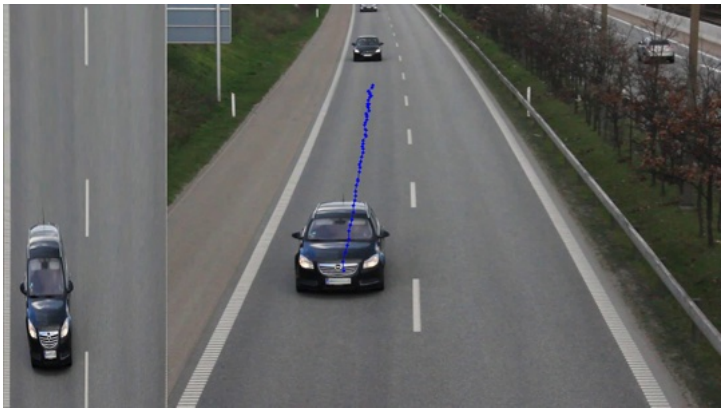


Figure 3.6: Example of a warped perspective (left), with the original real-world view (right), created from a homography.

Step 1 calculates the angles of the lines detected by the Hough transform relative to the x-axis. This represents the rotation of the lines.

Step 2 calculates the average angle of all the angles from Step 1. This represents the average rotation relative to the x-axis.

Step 3 calculates the slope of the average angle found in Step 2.

Step 4 finds the longest inner line from the Hough transform.

Step 5 creates two lines perpendicular to the average slope from Step 3 on the start and end position of the longest inner road line from Step 4.

Step 6 uses the two perpendicular lines to find four intersection points with the two outermost lines of the road. This is the first four points, of eight points in total for the homography, representing the real-world scene.

Step 7 calculates the number of road markings along the road by extracting the intensity of the pixels along the longest inner road line from Step 4 and doing a threshold on the intensity to estimate the number of groups. Figure 3.7 shows an example of pixel intensity along a set of road markings.

Step 8 calculates the length of the view by using national standards for the length of road markings (5 meters in Denmark [73]) and the space between them (10 meters in Denmark [73]).

Step 9 calculates the width of the view by multiplying the number of lanes with the national standard for lane width (3.75 meters in Denmark for wide highways [74]).

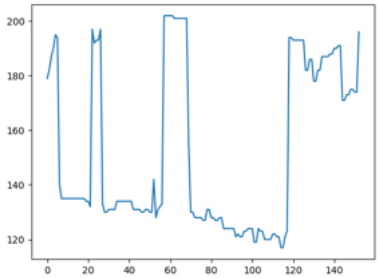


Figure 3.7: Example of pixel intensity along road markings on a highway.

Step 10 finds each corner of the rectangle with height as the road length and width as the road width, both multiplied by 100 for higher precision. These are now the second four points for the homography.

Step 11 estimates the homography with the four intersection points on the road from Step 6 and the four corner points from Step 10.

Step 12 determines the speed by transforming the tracking points of the vehicle to measurable points and measuring the distance the vehicle moved over a number of frames. Time is calculated using the frame rate from the video. The distance is converted to meters by dividing the pixel distances with 100 and the time is in seconds. Equation 3.1 calculates speed, using the first time a vehicle was seen (t_1), the first position (d_1), the last time seen (t_2) and the last position (d_2).

$$time = t_2 - t_1, \quad dist = abs(d_2 - d_1)/100, \quad speed = (dist/time) * 3.6 \quad (3.1)$$

3.1.6 License Plate Recognition

In Stage 6, once the system has recognized a vehicle and tracked it through its frames, it will be queued for license plate recognition when it leaves the frame. The system saves the area of the frame containing the vehicle for each frame and the biggest one is stored for further analysis. This ensures that the license plate is as big as possible for clearer recognition.

When the biggest image of the vehicle is found, it is stored in the system and a separate thread is spawned for further processing of the image. This makes the main system continue to operate with a high performance because the time consuming process of license plate recognition is offloaded to a separate thread and the main system can continue recognizing vehicles unhindered on the main thread.

The license plate recognition system is based on the open source automatic license plate recognition library OpenALPR 2.6.101 [54], which depends on the Tesseract OCR 3.0.4 library. The OpenALPR library is implemented in C++ with bindings for Python.

The process is a pipeline with several steps. First, it finds potential regions for license plates and converts them into black and white. It then searches for character-sized blobs and the edges of the license plate. The license plate is then deskewed before character segmentation and optical character recognition is performed. The last step is post-processing which outputs a list of possible matches based on confidence and performs a regular expression matching based on regional license plates. The regular expression matching checks for specific patterns consisting of letters and numbers. It first checks whether the license plate matches the Danish pattern of two letters and five number, before checking for Swedish license plates with three letters and three numbers. Additional checks can be implemented for other regions. The final chosen license plate is based on whether it matches a regional pattern and is returned to the main system on the main thread.

3.1.7 Privacy

Privacy is implemented in the video processing system in different stages to protect the identity of drivers. It is important that no identifiable data is available and the system ensures this in two important stages.

In Stage 4, which is the first time a vehicle is detected, no images or video of the vehicle is sent anywhere. The image of the vehicle is only stored locally for use in vehicle recognition and license plate recognition.

In Stage 7, the license plate is recognized. The image of the vehicle and the license plate is immediately deleted after being used and the license plate is hashed before being sent.

Video or images of vehicles never leave the video processing system and the license plates only leave the system as hashed values. Once the vehicle is registered and sent to the highway monitoring system, the data about it is deleted from the video processing system. From the perspective of the video processing system, data about a vehicle only exists the moment it is recognized but deleted shortly after.

3.2 Highway Monitoring System

The proposed highway monitoring system acts as a central server for every type of data collector, such as video processing systems using highway cameras to detect vehicles. In theory, any data collector can feed data into the system, such as a radar-based setup, as long as it detects vehicles, lanes and their speed. This section focuses on using the proposed video processing system as the input for the highway monitoring system.

When the system is setup initially, a centrally stored graph is constructed. The nodes of the graph are the different data collectors, for example, a node being a highway camera. The edges of the graph represents the road between two data collectors and is weighted by the distance between them.

The central server receives data from a data collector every time a vehicle is detected somewhere on the highways. The input contains a hashed identification number for the vehicle, the time it was detected, the lane it drove in, the speed it drove with, and the ID of the data collector.

Each time a vehicle is seen, the central highway graph is updated. Each node contains information about its own traffic, such as the number of vehicles and their speed. No individual vehicle data is stored in the graph and information about a vehicle is only stored for 20 minutes after it was last seen. The vehicle information is stored only to perform global speed detection between data collectors and vehicles are recognized based on the hashed license plate number. The 20-minutes window allow data collectors to be spaced far apart, e.g. 10 kilometers, and still track vehicles in slow moving traffic, while also preventing a large database of vehicle data to be extracted.

The system can output an overview of the traffic situation. This is done by constructing a weighted directed graph, with a node for each data collector. The edges of the graph are weighted by the current traffic situation, such as the proposed traffic indicator (see Section 3.2.1), current travel time, number of vehicles per hour or current average speed of vehicles. Figure 3.8 shows an overview of the proposed highway monitoring system.

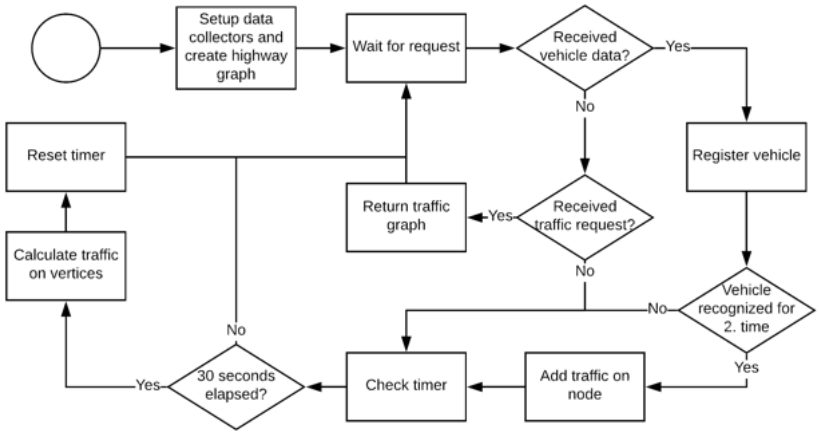


Figure 3.8: Overview of the highway monitoring system.

The system also estimates the average speed of vehicles. As opposed to the local speed detection, global speed detection uses multiple sightings of a vehicle to estimate its speed between two data collectors. Once a vehicle is detected for the second time, its average speed is calculated using the distance travelled and travel time between two data collectors. The average speed is useful for determining the general traffic situation and is used for estimating the traffic situation.

In order to reliably estimate the speed of vehicles, the cameras need to be synchronized to have the same central understanding of time. If the data collectors are not synchronized, the system is not able to accurately predict the travel time and speed of vehicles between two data collectors on the highway.

The proposed time synchronization is based on a simple sender to receiver synchronization. The highway monitoring system acts as the root node and handles the global time. Once a data collector, such as a highway camera, is set up, it sends a message to the system with its local timestamp. The main system can then calculate the offset to data collectors using the transmission time and the global time.

3.2.1 Traffic Detection

In order to provide traffic detection across several highways, the highway monitoring system depends on data from multiple data collectors. These provide data every time a vehicles passes by. The system can use this data to estimate how many vehicles are driving on different parts of the highway, which is represented by the number of vehicles per hour and their speed.

This information is used to calculate a traffic indicator value, which depends on the current load on the highway; the volume of vehicles compared to the capacity of the road and the general speed of the vehicles. The calculations for the road carrying capacity is based on the National Cooperative Highway Research Program's method for calculating road capacity [75], but speed is proposed as an additional parameter. The traffic indicator (T_I) is calculated in Equation 3.2, using the number of lanes on the road (N_{lanes}), the percentage of traffic consisting of trucks ($T_{\%}$), the road capacity in vehicles per hour (C_{vph}), the volume in vehicles per hour (V_{vph}), the volume-to-capacity (V_C) which compares the vehicle volume with the carrying capacity of the road, the average speed of vehicles on the road (S_{avg}), and the speed limit on the road (S_{limit}).

$$C_{vph} = \frac{2200 * N_{lanes}}{1 + T_{\%}/100}, \quad V_C = 1.2 - \frac{V_{vph}}{C_{vph}}, \quad T_I = 1 - (V_C * \frac{S_{avg}}{S_{limit}}) \tag{3.2}$$

A traffic indicator below 0.45 is considered light traffic, 0.45 to 0.70 is medium traffic, 0.70 to 0.90 is heavy traffic, and above 0.90 is extreme traffic.

The volume of vehicles is estimated every 30 seconds, which also give new calculations for the traffic indicator. This is to always keep an up to date view of the traffic situation, which is then never older than 30 seconds.

The traffic situation can be estimated at a specific point where a data collector is located, or it can be estimated on a section of the highway between two data collectors, using the average speed of vehicles between the two points and the vehicle volume going in and out of the section.

3.3 Simulator

A simple simulator program is proposed to simulate different traffic situations, in order to validate the highway monitoring system. Using the simulator it is possible to set a number of parameters for the simulation, such as the inflow of vehicles per hour, the average speed of the vehicles, the number of lanes on the highway, the percentage of the vehicles that are trucks and whether a single point or two on the highway are simulated.

Inflow of Vehicles per Hour determines the number of vehicles spawned and when they are "seen" at the first data collection point, such as a simulated highway camera. The vehicles are spaced out over an hour and are spawned at every interval as long as the simulator runs. For example, if the inflow is 1000 vehicles per hour, a vehicle will spawn every 3.6 seconds.

Average Speed of Vehicles determines the general speed. The speed of individual vehicles is determined by picking a random speed from a Gaussian distribution, with the average speed being the mean value and standard deviation of 5. Vehicles also have a middle and end speed if two points on the highway are simulated, which makes it possible to simulate vehicles between the two data collection points.

Number of Lanes on the Highway is used to simulate multi-lane highways and simulated vehicles might switch lanes to get around slower vehicles. The lane of a vehicle is determined by whether the vehicle would catch up on another vehicle in front of it with its current speed. A vehicle that cannot continue its speed switches to the next lane.

Percentage of the Vehicles That Are Trucks is relevant for a more life-like simulation. The trucks generally drive slower and a vehicle must sometimes switch to another lane to continue its speed.

Single Point or Two on the Highway is chosen to determine whether vehicles are seen only once or at two consecutive connected data collection points. This is useful for testing the global speed detection since two separate speed measurements from two different points on the highway are simulated.

4
Analysis

Chapter 4 presents the analysis and results of the video processing system and highway monitoring system proposed in Chapter 3. The evaluation of the systems focus on their performance and accuracy in detection.

Section 4.1 presents the analysis of the proposed video processing system's performance. The performance is evaluated using a framerate counter in the video processing system, which measures the number of frames that can be processed by the system every second.

First, the baseline performance is measured, before determining the performance of the video processing system. This is done by counting the number of frames the system can read from a video per second without any processing. The system's performance is compared to the baseline performance.

The section also presents the video processing system's accuracy in detection and recognition of vehicles, lanes, and license plates, using different validation methods.

Section 4.2 presents the analysis of the proposed highway monitoring system's performance. The section focus on the detection of traffic situations and the performance of the traffic indicator proposed in Chapter 3.

The traffic indicator is evaluated using a real-world traffic situation from the video processing system and different other simulated traffic situations from the proposed simulator.

4.1 Video Processing System

This section evaluates the proposed video processing system using a low-quality highway video with a resolution of 1920 × 1080 pixels [76].

4.1.1 Performance

The video processing system's performance is measured using a framerate counter, which counts the number of frames that can be handled by the system per second.

In order to generalize the performance of the system, the evaluation is done on two different hardware setups. The first hardware setup has a 4 GHz 6th-generation Intel Core i7-6700K CPU and is running Windows 10. The other hardware setup has a 2.5 GHz 4th-generation Intel Core i7-4870HQ CPU and is running MacOS 10.14. Both hardware setups load the video from a Solid State Drive (SSD) and run the system on the CPU.

Baseline Performance is used to compare the video processing system's performance to the general raw performance. The baseline is determined by running the video unprocessed on the two hardware setups and measuring the frames per second. The baseline performance is 162 FPS on the i7-6700K CPU and 208 FPS on the i7-4870HQ CPU. Table 4.1 shows the baseline performance on the two hardware setups.

CPU	**SSD**	**OS**	**Baseline**
i7-6700K	Samsung SM951	Windows 10 (1803)	162 fps
i7-4870HQ	Apple SM0512G	MacOS 10.14.1	208 fps

Table 4.1: Baseline performance for running an unprocessed video on two different hardware setups.

It can be seen that the baseline performance is very high. The minimum performance the video processing system needs to achieve to be reliable is 30 FPS, while the recommended performance is at least 60 FPS.

Every stage of the system impacts the performance of the system. The baseline performance is the system's performance before any stages.

Stage 0 is the baseline performance at 162 FPS on the i7-6700K and 208 FPS on the i7-4870HQ.

Stage 1 stabilizes the background image and extracts the foreground. This reduces the framerate to 138 FPS on the i7-6700K and 173 FPS on the i7-4870HQ. The performance is worst in the beginning of the video until the background is stabilized. This is the first high impact on the performance.

Stage 2 detects the lines on the background image and estimates the lanes from the lines. It reduces the framerate to 112 FPS on the i7-6700K and 124 FPS on the i7-4870HQ. This is the second high impact on the performance.

Stage 3 extracts detected blobs from the foreground. It tracks vehicles through frames and reduces the framerate to 111 FPS on the i7-6700K and 122 FPS on the i7-4870HQ.

Stage 4 recognizes vehicles by classifying new blobs. It reduces the framerate to 108 FPS on the i7-6700K and 120 FPS on the i7-4870HQ.

Stage 5 estimates the homography for the road and detects the speed of vehicles. This reduces the framerate to 107 FPS on the i7-6700K and 118 FPS on the i7-4870HQ.

Stage 6 spawns new threads for license plate recognition when vehicles leave the frame. It reduces the framerate to 103 FPS on the i7-6700K and 114 FPS on the i7-4870HQ.

Figure A.1-A.7 in Appendix A show the frames per second for each stage. It is seen that the worst performance of each stage is at the beginning of the video. This is because the background is stabilized first. Once it is stable the system runs with a higher framerate, that variate little. This means that the system needs more time when setting up or when the view changes and the background needs to be stabilized again. It does not affect the system once the background model is established. It only impacts the performance when the video processing system is turned on or the view changes.

Even though both hardware setups start with a very different framerate for running the system, they run the final system at almost the same framerate: A 36% reduction from 162 FPS to 103 FPS on the i7-6700K and a 45% reduction from 208 FPS to 114 FPS on the i7-4870HQ. Table 4.2 and Figure 4.1 shows the impact on the performance for each stage in the system.

Stage	**Description**	**i7-6700K**	**i7-4870HQ**
Stage 0	Baseline	162 FPS	208 FPS
Stage 1	Background Subtraction	138 FPS	173 FPS
Stage 2	Lane Detection	112 FPS	124 FPS
Stage 3	Blob Detection	111 FPS	122 FPS
Stage 4	Vehicle Recognition	108 FPS	120 FPS
Stage 5	Speed Detection	107 FPS	118 FPS
Stage 6	License Plate Recognition	103 FPS	114 FPS

Table 4.2: Performance of the video processing system through the stages.

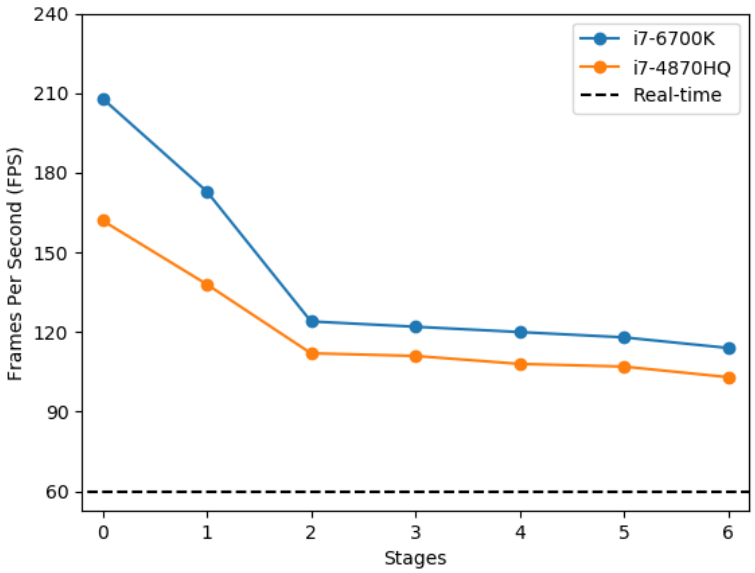


Figure 4.1: Performance of the video processing system through the stages.

4.1.2 Performance of Components

Each component within the video processing system must perform with reasonable accuracy for the overall system to reliably detect and recognize vehicles. The components are tested to see how accurate they are in different situations and the results are analyzed in the context of general highway situations. The highway video is overlayed with the results of detection [77].

4.1.2.1 Lane Detection

Lane detection should be robust to different rotations of the road. No matter which direction the road is facing, the lane detection should be able to detect the lines and correctly estimate the polygons of the lanes. In order to evaluate the accuracy of lane detection, geometric transformations are applied to the image. Eight rotations of the background image are presented to the system. Table 4.3 shows the detection results for the eight different road directions. "Detected angle" represents the direction of the lines counted in degrees from north or straight up on the image, "Points" represents how many of the four needed points are detected, and "Lanes" represents the lanes detected on the 2-lane highway used for testing. It is seen that no matter the direction of the road, the lane detection picks up the number of lanes without problems.

Direction of road	**Detected angle**	**Points**	**Lanes**
North	2.56°	4	2
North-East	51.94°	4	2
East	93.99°	4	2
South-East	138.48°	4	2
South	182.56°	4	2
South-West	231.97°	4	2
West	272.10°	4	2
North-West	308.02°	4	2

Table 4.3: Result of lane detection on 8 different road direction.

4.1.2.2 Vehicle Recognition

The neural network, implemented as a multi-layer perceptron (see Chapter 3), is responsible for predicting whether a specific blob is a vehicle or not.

Using Principal Component Analysis (PCA) to reduce the features to 100 dimensions, shows that this accounts for about 70% of the variations in the extracted features. This can be further reduced using t-Distributed Stochastic Neighbor Embedding (t-SNE) to two-dimensional data. The two-dimensional data can then be visualized and a cluster of vehicles can be distinguished. Figure 4.2 shows the data using dimensionality reduction.

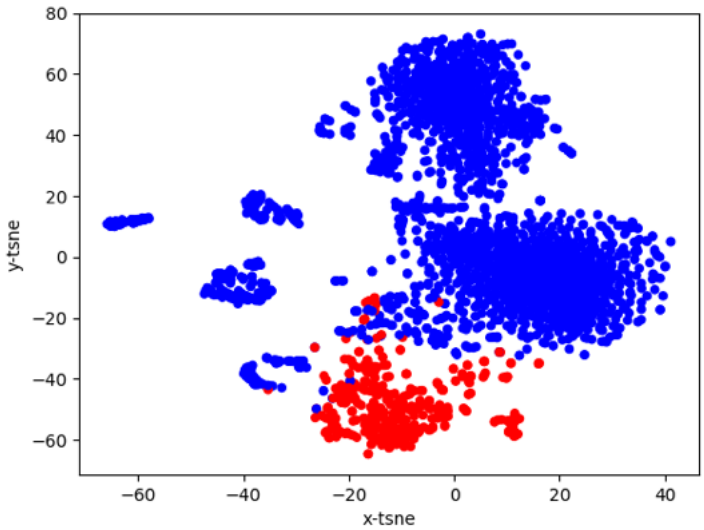


Figure 4.2: Visualized data with vehicles (red) and non-vehicles (blue).

The accuracy of the neural network for detecting vehicles is 93%, while the accuracy for detecting non-vehicles is 98%. This gives an overall accuracy of 97%. Splitting the dataset on 40% for test and using the test data, the neural network detects 675 images as non-vehicles and 139 images as vehicles, while it predicts 8 false positives and 8 false negatives. Table 4.4 shows the results of testing the machine learning model.

	Precision	**Recall**	**F1-score**
non-vehicle	0.98	0.99	0.98
vehicle	0.94	0.91	0.93
	0.97	0.97	0.97

Table 4.4: Accuracy of multi-layer perceptron for vehicle detection [22].

Measuring the performance of predicting vehicles shows that a single prediction takes around 0.05 ms per image [22]. It is clear that using the multi-layer perceptron is a great choice both when it comes to accuracy of vehicle detection but also when it comes to the general performance of the system, as it provides a low impact on the time it takes to process a frame. When combining the time it takes to extract HOG features from a single image (0.08 ms/image) and the time it takes to classify the image (0.05 ms/image), it is seen that it only has a minor impact on the general performance [22].

4.1.2.3 Speed Detection

Performing speed detection on the highway video [76] yields an average speed of 98 km/h. The speed is realistic based on the speed limit, but further evaluations is needed to validate. Listing B.1 under Appendix B shows the output of the system running the video.

In order to further evaluate the proposed speed detection method, geometric transformations can be applied to the original video, such as rotation.

Rotating the highway video [76] 180 degrees and running it through the video processing system yields almost similar results as the original video. The vehicles in the rotated video get their average speed detected to 83.5 km/h, while the original video had 98 km/h. Even though the video was rotated, not vehicles driving away from the camera, the system still managed to get an acceptable result close to the original. Listing B.4 under Appendix B shows the output of the system running the rotated video.

4.1.2.4 License Plate Recognition

The recognition of license plates is essential for recognizing previously seen vehicles. Given a dataset of 100 images of vehicle license plates [78], the system can correctly recognize 78 license plates from different angles. 22 license plates were wrong because the predicted license number was wrong or because the license plate was not detected at all. The system obtained a 78% accuracy on the tested dataset.

Running license plate recognition on the highway video [76], the system can recognize 15 license plates out of 17. The output from the video processing system can be seen in Listing B.1 under Appendix B.

Neither the video nor the image dataset contains license plates captured at night, in rain, or in snow. During such times license plate recognition would not work based on computer vision without additional light.

Once the system is ready to recognize a vehicle's license plate, a new thread is spawned. It is essential to look at the performance impact this has on the system and the machine the system is running on. Running license plate recognition on the main thread have a higher impact on the performance. The system runs at 73 FPS when recognizing license plates on the main thread compared with 103 FPS on the i7-6700K. This is because the system waits until a license plate is recognized before it continues with the next frame. The process of recognizing license plates takes up to 1 second for each license plate, with around 0.4 seconds being the average. Threads are therefore essential for the system's performance.

Considering a situation where two vehicles arrive every second with a worst-case scenario of 2 seconds per license plate recognized. The system spawns two threads after the first second and two new after the next second. On the third second the system is done with the first two threads and spawns two new threads. In this situation, the system runs with at most four separate threads. Given a general usage of 100 MB of RAM per thread, the system would need at minimum 512 MB of RAM for the system to run.

4.2 Highway Monitoring System

The proposed traffic indicator depends on vehicles per hour and their average speed. According to the current model of the traffic indicator, a high number of vehicles might not be a problem as long as the speed is also high. The real problem is when speed decreases and vehicles per hour increases. A high number of vehicles with a low speed is often a sign of traffic jams. Vehicles per hour will generally decrease if the speed decreases, unless more vehicles driver closer to each other.

The whole spectrum is seen when calculating the proposed traffic indicator for different speeds and vehicles per hour. Figure 4.3 shows how the proposed traffic indicator depends on vehicles per hour and speed on a 2-lane highway, where green represents light traffic and red represents heavy traffic. The traffic indicator is a decimal number between 0 and 1.

Applying the proposed traffic indicator to the highway video used for testing the video processing system [76], gives an indication about the traffic situation in that specific video. The traffic situation in the video is 1020 vehicles per hour with an average vehicle speed of 98 km/h. The traffic indicator is calculated to 0.16 for the video, which means a light traffic situation (below 0.45). This seems accurate based on the traffic in the video, which seems to pose no risk of an elevation in the traffic situation.

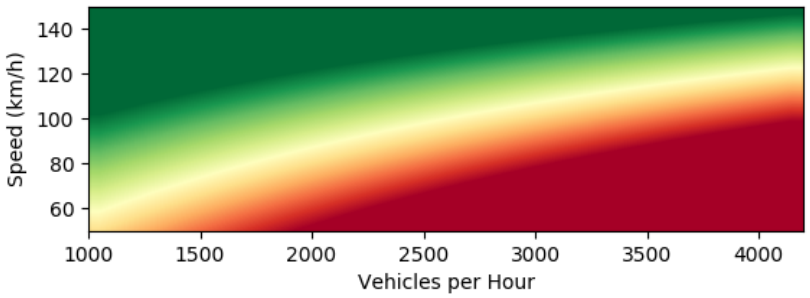


Figure 4.3: Traffic indicator depends on vehicles per hour and average speed.

4.2.1 Simulated traffic data

Using the simulator, different traffic situations can be estimated. This is useful for testing the highway monitoring system's response on different data and its predictions about the traffic.

Table 4.5 shows the traffic indicators for different traffic situations. The first example simulates the traffic situation from the recorded highway video [76]. Listing B.2 under Appendix B shows the output from the simulator and can be compared to the video processing system output in Listing B.1 also under Appendix B.

Lanes	**Vehicles/hour**	**Avg. Speed**	**Indicator**	**Traffic**
2	1020	98	0.16	Light
2	2640	100	0.51	Medium
2	3590	80	0.78	Heavy
2	4210	105	0.86	Heavy
3	2610	100	0.30	Light
3	3590	80	0.56	Medium
3	4210	105	0.52	Medium

Table 4.5: The predicted traffic based on different traffic situations from the simulator.

Looking at the predictions from the simulated data, it is seen that the system depends on both vehicles and their speed in order to determine the traffic indicator.

It is also seen that the number of lanes is important, since it determines the capacity of the road. More lanes means the road is less likely to be saturated by a high number of vehicles per hour.

Using this information it is possible to predict traffic situations. With an update interval of 30 seconds, traffic situation changes can be monitored, such as when the vehicle volume increases or their speed decreases. The system can predict an increase in the severity of the traffic situation.

5
Discussion

Chapter 5 discusses the proposed systems in general and its limitations, while also discussing hardware requirements for the system and its privacy. At last future works is discussed.

Section 5.1 discusses the hardware requirements for the proposed video processing system. It also discusses the performance-based findings from the analysis in Chapter 4 and the future requirements for deployment.

Section 5.2 discusses the privacy aspects of monitoring systems. It also discusses the general privacy in the proposed systems and the extend of the privacy implementation in the video processing system.

Section 5.3 discusses the future work that can be carried out for this project and which direction it could take in the future.

The system proposed and built in this thesis provides high performance on low-cost hardware while providing the same functionality as other systems. Most highway monitoring systems support a similar set of features as the one proposed in this thesis. The system use many of the image processing and machine learning methods proposed in this thesis and if not, another approach is used to obtain a similar result. However, compared to other systems and other research studies done, this thesis proposes a system, with the same level of performance and accuracy as other systems, while also protecting the privacy of drivers. No functionality or performance was sacrificed to obtain privacy. It was built in from the beginning, and the system still performs on the level of other systems. Privacy needs to be a standard for future systems, especially monitoring systems that allow mass surveillance.

In general, automated traffic monitoring is vital for the future of transport, but the technology faces more challenges, other than privacy. The challenges depend on the specific method chosen, such as whether to use a camera-based system or a radar-based systems.

This thesis proposed a camera-based setup. The biggest issue with a camera-based setup is the lack of vision during night or vision-impairing weather such as rain or snow. The lousy visibility hinders the detection and recognition. License plates cannot be detected, and lanes could be a problem as well if snow or rain covers or blurs the road markings. If the image processing system already knows the lines, have estimated the homography for the road, and the camera did not move, it can still predict lanes and speed as long as vehicles can be seen and detected.

Another challenge is the change of lighting by the sun. The background subtractor needs to be very robust to changes in order to always pick out the foreground from the background. Strong sun can also impact the performance because of shadows. If a car shadow is long and visible it might get picked up as part of the foreground object and can decrease the accuracy of vehicle detection.

There is many ways to describe traffic, such as travel time or volume of vehicles compared to the road capacity. Travel time is a popular choice often used in map applications, such as Google Maps. In this thesis, a traffic indicator is proposed as the primary way to inform about traffic. The limitation of using the volume of vehicles is that it does not take into account the actual flow of vehicles. In order to solve this, the traffic indicator proposed by this thesis uses speed as a parameter that affects the overall traffic indicator. Because no universal way to describe traffic exists, the system can output any available information, such as travel time, the proposed traffic indicator, or vehicle volume. The type depends on the request, at which point the system calculates the traffic on the graph and returns the result.

5.1 Hardware

The highway monitoring system accepts any data from any data collector, but this thesis proposes a video processing system. A video processing system needs a camera that records video in a sufficiently high resolution. This is important for license plate recognition, which requires detecting letters and numbers of license plates. The resolution depends on the distance to vehicles and other factors in the placement of the camera.

Looking at the performance of the system, it is remarkable that the system is running faster on the older and less powerful mobile processor i7-4870HQ than it does on the newer desktop processor i7-6700K. It might have something to do with the operating systems and the implementation of Python on the two platforms. It could also be because more background processes were running on the Windows system than on the MacOS system in this specific case. The situation changes, however, if the machine shows the processed video and the results, at which point the desktop system greatly outperforms the laptop system, because of its graphics processor.

If the system were deployed on a real highway, it would be essential to choose a low-cost CPU, which would still be powerful enough to run the system. Relevant processors include mobile processors with combined CPU and GPU, such as the Nvidia Tegra X1 chip, which is a powerful option. This would make the system able to process the video in real-time, while still keeping a low hardware cost per camera.

The system also needs a way for the different data collectors to communicate with the central server or main system. It might not be possible to get internet using cables to the different collectors. Therefore a wireless solution could be needed for connection. Many wireless technologies are useful, such as city-wide wireless technology, e.g. LoRa or Sigfox. Such options provide a low-energy and low-cost option. Other options in the future might include using 5G to connect data collectors to the highway monitoring system.

5.2 Privacy

In order to protect the privacy of highway drivers, the system sends no video or images of vehicles or drivers anywhere. Once an image is processed by the individual camera it is removed from the proposed system. The image of a vehicle is only used locally for vehicle and license plate recognition.

Another important feature is that license plates are hashed before they left the video processing system and deleted from the main system after 20 minutes of not being seen. The reason for hashing the license and sending it to the proposed highway monitoring system, is that it allows the system to detect the average speed between data collectors. Another reason, perhaps more important, is that law enforcers can ask the system for a specific license, without needing to know any license in the system. With this approach, law enforcers can upload a hashed license to the system, and the system can respond the next time the license plate is seen. Law enforcers cannot, however, get any previous data about the vehicle nor can they get information about other vehicles. Keeping a hashed list of vehicle licenses from the past 20 minutes ensures a high level of privacy. First, a malicious party cannot extract a huge database of licenses, because license plates only exist as hashed values. Second, should a malicious party be able to extract said hashed values, then only the last 20 minutes is available and not days, weeks or months of data.

Privacy is often a discussion about how much data should be available to law enforcers and how much should be kept private. This thesis proposes a middle way, which protects the identity of drivers, but still allow law enforcers to search out specific individuals, possibly with a court order. This makes it challenging to make mass surveillance because the system cannot be asked for every license plate and it cannot be retroactively searched for vehicles seen in the past.

5.3 Future Work

This thesis proposes a coherent highway monitoring system and achieves great results, but further work can be carried out in this project to validate the claims for the system's performance with new evaluation methods.

In this thesis, a performance-optimized background subtractor is proposed, but future research would need to investigate a background subtractor that handles change of lightning and car shadows. This would impact the performance more than the proposed method, but with the system's great performance it should not decrease the framerate much.

An important next step is the validation of speed detection. Using additional equipment the video processing system must be calibrated to measure the correct speed. This could be done using a radar-based setup.

Future work should look at a more versatile simulator to simulate more realistic and different situations. It could look into the Intelligent-Driver Model [79][80], with accelerations and braking decelerations of the drivers.

More work should also be done on more highway videos to validate the system in more situations and specific cases. Such videos should also include different weather situations, such as rain and snow, and time of day, such as sunset, sunrise, and night. Videos should also include different camera perspectives to further validate the claims for lane and speed detection.

Another important future step is to investigate the possibilities for different types of data collectors to work together in the same highway monitoring system. The proposed highway monitoring system in this thesis handles data from any data collector, not just the proposed video processing system. It would be beneficial to look at different methods for gathering data to see how they compare and how the highway monitoring system can predict data from different sources. Such a data collector could be a radar-based setup, which does not suffer the same limitations from weather and lighting. Other types of data collectors could also be identified in future work and could provide new benefits to the system.

6
Conclusion

The purpose of this thesis is to uncover the design and creation of a highway monitoring system. The requirements are to obtain high performance and accuracy on a privacy-focused system.

The system obtains the expected accuracy for most of its components, above the expected 85%, with most being around 90% or higher. The only component that does not obtains the needed accuracy in its extensive test is the license plate recognition component, which performs at an accuracy of 78%. It does, however, detect 88% of license plates in the recorded highway video [76]. The system needs more training and calibration for the detection and recognition methods to obtain higher results.

Running the system on two different hardware setups shows that the system can run in real-time on a less powerful four-year-old mobile processor and a two-year-old desktop processor. It does so with a significant margin, a performance of around 100 FPS, where at least 30 FPS are required. With this performance, it would not be a problem to run the system, even on a camera-based setup, with a framerate of 60 FPS.

Data collectors and the highway monitoring system is separated and no sensitive information is sent between them. The system has no identifiable data about drivers and no images or videos are sent anywhere. License plate numbers are hashed before leaving the video processing system.

It is seen that a highway monitoring system can be designed and built with privacy in mind, without hindering the performance or accuracy of the system.

Bibliography

[1] Danmarks Statistik, "Statistisk analyse af vejtransport," 2015.

[2] Vejdirektoratet, "Hvordan udvikler trafikken sig?." http://www.vejdirektoratet.dk/DA/viden_og_data/statistik/trafikken%20i%20tal/hvordan_udvikler_trafikken_sig/Sider/default.aspx, Online; accessed 1-January-2019.

[3] Vejdirektoratet, "Trafikcentret." http://www.vejdirektoratet.dk/DA/trafik/bagom/trafikinformationscentret/Sider/default.aspx, Online; accessed 1-January-2019.

[4] European Commission, "Intelligent transport systems." https://ec.europa.eu/transport/themes/its_en, Online, accessed 1-January-2019.

[5] R.-Y. Wang, Y.-T. Chuang, and C.-W. Yi, "A crowdsourcing-based road anomaly classication system," *18th Asia-Pacific Network Operations and Management Symposium*, 2016.

[6] F. J. Villanueva, J. Albusac, L. Jiménez, D. Villa, and J. C. López, "Architecture for smart highway real time monitoring," *27th International Conference on Advanced Information Networking and Applications Workshops*, 2013.

[7] X. Chen and Q. Meng, "Robust vehicle tracking and detection from UAVs," *IEEE*, 2015.

[8] Z. Lei, Z. Xue-fei, and L. Yin-ping, "Research of the real-time detection of traffic flow based on OpenCV," *International Conference on Computer Science and Software Engineering*, 2008.

[9] D. Li, B. Liang, and W. Zhang, "Real-time moving vehicle detection, tracking, and counting system implemented with OpenCV," *4th IEEE International Conference on Information Science and Technology*, 2014.

[10] M. Soleh, G. Jati, A. T. Sasongko, W. Jatmiko, and M. H. Hilman, "A real time vehicle counting based on adaptive tracking approach for highway videos," *International Workshop on Big Data and Information Security*, 2017.

[11] H. Unno, K. Ojima, K. Hayashibe, and H. Saji, "Vehicle motion tracking using symmetry of vehicle and background subtraction," *IEEE Intelligent Vehicles Symposium*, 2007.

[12] S. Miyata and K. Oka, "Automated license plate detection using a support vector machine," *14th International Conference on Control, Automation, Robotics and Vision*, 2016.

[13] X. He and D. Zeng, "Real-time pedestrian warning system on highway using deep learning methods," *International Symposium on Intelligent Signal Processing and Communication Systems*, 2017.

[14] Z. Hui, X. Yaohua, M. Lu, and F. Jiansheng, "Vision-based real-time traffic accident detection," *Proceeding of the 11th World Congress on Intelligent Control and Automation*, 2014.

[15] A. Jacobsen and L. Bohr, "A quick overview of methods for car detection," *IT-University of Copenhagen*, 2018.

[16] N. Lu, J. Wang, Q. Wu, and L. Yang, "An improved motion detection method for real-time surveillance," *International Journal of Computer Science*, 2008.

[17] Y. Du, C. Liu, and W. Shi, "A real-time vehicle recognition method based on video sequence images," *9th International Conference on Electronic Measurement Instruments*, 2009.

[18] M. Djalalov, H. Nisar, Y. Salih, and A. S. Malik, "An algorithm for vehicle detection and tracking," *International Conference on Intelligent and Advanced Systems*, 2010.

[19] A. B. Godbehere, A. Matsukawa, and K. Goldberg, "Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation," *American Control Conference*, 2012.

[20] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," *International Conference on Pattern Recognition*, 2004.

[21] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, 2006.

[22] A. Jacobsen and L. Bohr, "Detecting moving vehicles in real-time," *IT-University of Copenhagen*, 2018.

[23] S. Zeeve, "The fastest opencv background subtraction." `https://sagi-z.github.io/BackgroundSubtractorCNT/`, Online; accessed 1-January-2019.

[24] B. Girod, "Morphological image processing," *Stanford University*, 2013.

[25] H. Samet and M. Tamminen, "Efficient component labeling of images of arbitrary dimension represented by linear bintrees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1988.

[26] N. U. Rashid, N. C. Mithun, and B. R. Joy, "Detection and classification of vehicles from a video using time-spatial image," 2010.

[27] B. F. Wu and J. H. Juang, "Adaptive vehicle detector approach for complex environments," 2012.

[28] deepai.org, "Feature extraction." https://deepai.org/machine-learning-glossary-and-terms/feature-extraction, Online, accessed 1-January-2019.

[29] C. Harris and M. Stephens, "A combined corner and edge detector," *Alvey Vision Conference*, 1988.

[30] N. Dala and B. Triggs, "Histograms of oriented gradients for human detection," 2005.

[31] Mithi, "Vehicle detection with HOG and linear SVM." https://towardsdatascience.com/vehicles-tracking-with-hog-and-linear-svm-c9f27eaf521a, Online; accessed 1-January-2019.

[32] C. M. Bishop, "Pattern recognition and machine learning," *Springer*, 2006.

[33] V. Vapnik and A. Lerner, "Pattern recognition using generalized portrait method," *Automation and Remote Control*, 1963.

[34] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers," 1992.

[35] M. van Gerven and S. Bohte, "Artifical neural networks as models of neural information processing," *Frontiers in Computational Neuroscience*, 2018.

[36] H. Shi, Z. Lui, Y. Fan, X. Wang, and T. Huang, "Effective object detection from traffic camera videos," *Image Formation and Processing (IFP) Group, University of Illinois*, 2017.

[37] H. Maciejewski, J. Mazurkiewicz, K. Skowron, and T. Walkowiak, "Neural networks for vehicle recognition," 1997.

[38] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.

[39] R. Girshick, "Fast R-CNN," *Microsoft Research*, 2015.

[40] S. Ren, K. He, R. Girshick, , and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," 2016.

[41] J. Redmon, S. K. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2015.

[42] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," 2016.

[43] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018.

[44] J. Redmon, "YOLO: Real-time object detection." `https://pjreddie.com/darknet/yolo/`, Online; accessed 1-January-2019.

[45] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Computer Science Department, Carnegie-Mellon University*, 1981.

[46] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," *Computer Vision Laboratory, Linköping University*, 2003.

[47] J. W. Baek, B.-G. Han, H. Kang, Y. Chung, and S.-I. Lee, "Fast and reliable tracking algorithm for on-road vehicle detection systems," *ICUFN*, 2009.

[48] F. Dellaert and C. Thorpe, "Robust car tracking using kalman filtering and bayesian templates," *Conference on Intelligent Transportation Systems*, 1997.

[49] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (alpr): A state-of-the-art review," *IEEE Transactions on Circuits and Systems for Video Technology*, 2013.

[50] T. Agui, H. J. Choi, and M. Nakajima, "An extracting method of car number plate by image processing," *IEICE*, 1987.

[51] S. Gael and S. Dabas, "Vehicle registration plate recognition system using template matching," *IEEE*, 2013.

[52] V. Karthikeyan, R. Sindhu, K. Anusha, and D. S. Vijith, "Vehicle license plate character segmentation - a study," 2013.

[53] L. Xiaojbo, L. Xiaojinh, and H. Wei, "Vehicle license plate character recognition," *International Conference Neural Networks and Signal Processing*, 2003.

[54] OpenALPR Technology, "Automatic license plate recognition." `http://www.openalpr.com/`, Online; accessed 1-January-2019.

[55] A. Key, "Tesseract: an open-source optical character recognition engine." `https://www.linuxjournal.com/article/9676`, Online; accessed 1-January-2019.

[56] A. Tungkastan, N. Jongsawat, and W. Premchaiswadi, "A proposed framework for automated real-time detection of suspicious vehicles," *International Conference on ICT and Knowledge Engineering*, 2014.

[57] D. F. Llorca, R. Arroyo, and M. A. Sotelo, "Vehicle logo recognition in traffic images using hog features and svm," *16th International IEEE Conference on Intelligent Transportation Systems*, 2016.

[58] X. Wei, Z. Zhang, Z. Chai, and W. Feng, "Research on lane detection and tracking algorithm based on improved hough transform," *the International Conference of Intelligent Robotic and Control Engineering*, 2018.

[59] C. Y. Low, H. Zamzuri, and S. A. Mazlan, "Simple robust road lane detection algorithm," *IEEE*, 2014.

[60] H. Jang, S. Baek, and S. Park, "Curved lane detection using robust feature extraction," *The 2nd International Conference on Systems and Informatics*, 2014.

[61] J. Gerát, D. Sopiak, M. Oravec, and J. Pavlovicová, "Vehicle speed detection from camera stream using image processing methods," 2017.

[62] M. A. Alavianmehr, "A new vehicle detect method based on gaussian mixture model along with estimate moment velocity using optical flow," 2014.

[63] K. K. Kumar, P. Chandrakant, S. Kumar, and K. Kushal, "Vehicle speed detection in video frames using corner detection," *Fifth International Conference on Signal and Image Processing*, 2014.

[64] X. Pan, Y. Guo, and A. Men, "Traffic surveillance system for vehicle flow detection," *Second International Conference on Computer Modeling and Simulation*, 2010.

[65] T. Tahmid and E. Hossain, "Density based smart traffic control system using canny edge detection algorithm for congregating traffic information," *3rd International Conference on Electrical Information and Communication Technology*, 2017.

[66] X. Lei, W. Qing, C. Xiumin, W. Jun, and C. Ping, "Traffic jam detection based on corner feature of background scene in video-based its," *IEEE International Conference on Networking, Sensing and Control*, 2008.

[67] X. Wang, L.-M. Meng, B. Zhang, J. Lu, and K.-L. Du, "A video-based traffic violation detection system," *International Conference on Mechatronic Sciences, Electric Engineering and Computer*, 2013.

[68] H. Xie, L. Kulik, and E. Tanin, "Privacy-aware traffic monitoring," *IEEE Transactions on Intelligent Transportation Systems*, 2010.

[69] R. Shigetomi, M. Sato, K. Uehara, A. Otsuka, H. Sunahara, and H. Imai, "An efficient scheme to protect privacy in probe vehicle information system," *8th International Conference on ITS Telecommunications*, 2008.

[70] A. Cavoukian, "Privacy by design – the 7 foundational principles," *Privacy and Big Data Institute*, 2009.

[71] A. Kung, J. Freytag, and F. Kargl, "Privacy-by-design in its applications," *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2011.

[72] F. B. Narcizo, "Dataset for the Introduction to Image Analysis and Machine Learning course - spring 2018," *IT University of Copenhagen*, 2018.

[73] Vejregler, "Afmærkning på kørebanen, dimensioner." `http://www.vej-eu.dk/Net_Kursusmappe/ID15473/7%20Vejregler/5.6%20Dimensioner%20maj%202013HF_i.pdf`, 2013.

[74] Vejdirektoratet, "Motorvejstværprofil," 2012.

[75] R. Dowling, P. Ryus, B. Schroeder, M. Kyte, F. Thomas Creasey, N. Rouphail, A. Hajbabaie, and D. Rhoades, "Planning and preliminary engineering applications guide to the highway capacity manual," *NCHRP Report 825*, 2016.

[76] F. B. Narcizo, "Recording of vehicles on a highway for the Introduction to Image Analysis and Machine Learning course - spring 2018," *IT University of Copenhagen*, `https://youtu.be/RH8fuOvL4DU`, 2018.

[77] F. B. Narcizo and A. Jacobsen, "Detection of vehicles on a highway adapted from the recording of vehicles on a highway for the Introduction to Image Analysis and Machine Learning course - spring 2018," *IT University of Copenhagen*, `https://youtu.be/tTR0iVVrqEM`, 2018.

[78] A. Jacobsen, "Dataset of different cars collected from websites selling used cars," 2018.

[79] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," 2000.

[80] M. Treiber and A. Kesting, "Traffic flow dynamics - data, models and simulation," *Springer*, 2013.

Appendices

A

Figures

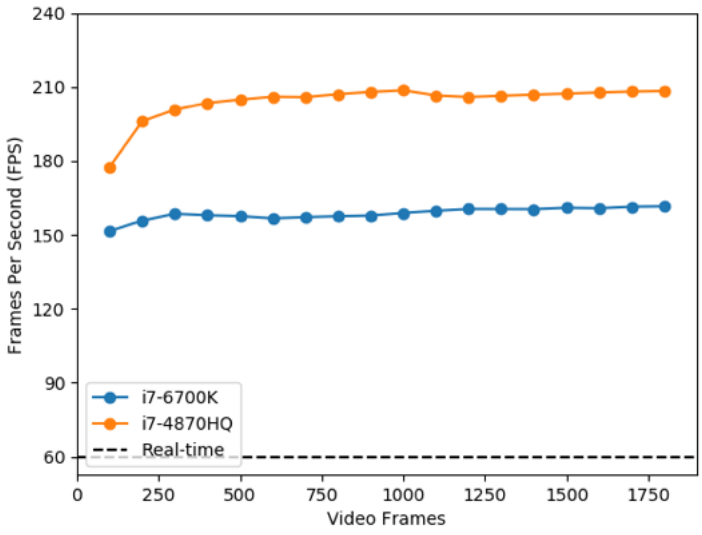


Figure A.1: Performance of the video processor without any processing.

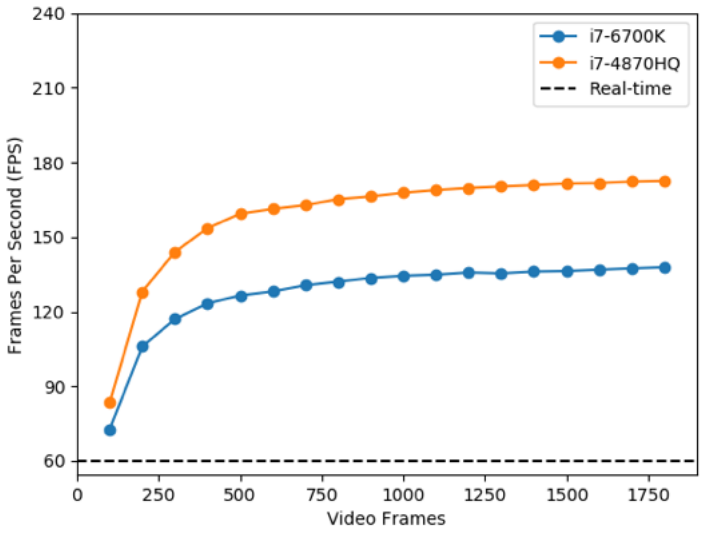


Figure A.2: Performance of the video processing system after Stage 1.

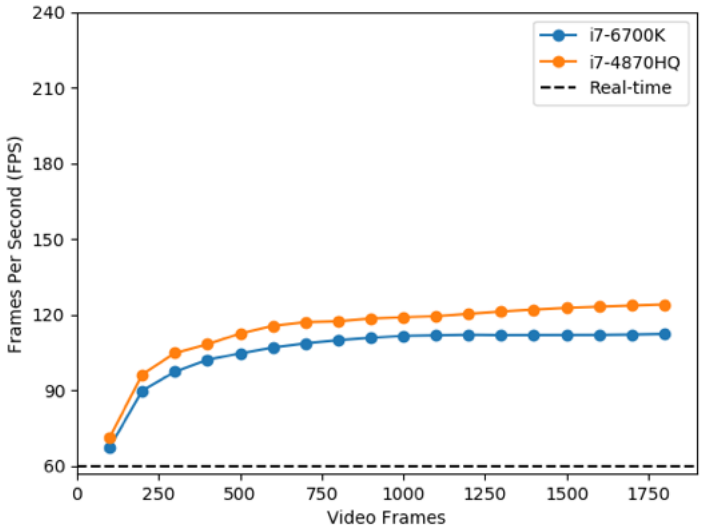


Figure A.3: Performance of the video processing system after Stage 2.

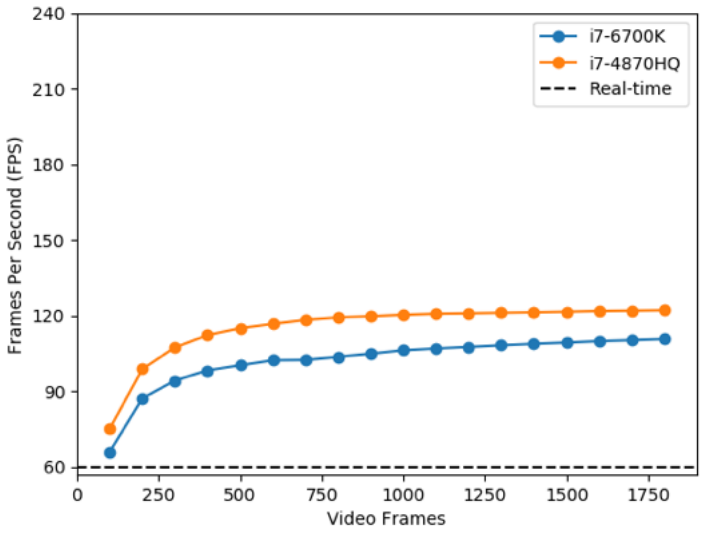


Figure A.4: Performance of the video processing system after Stage 3.

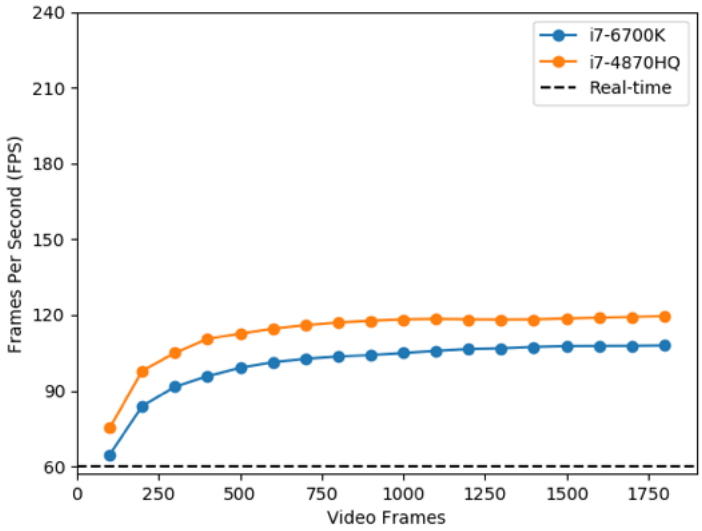


Figure A.5: Performance of the video processing system after Stage 4.

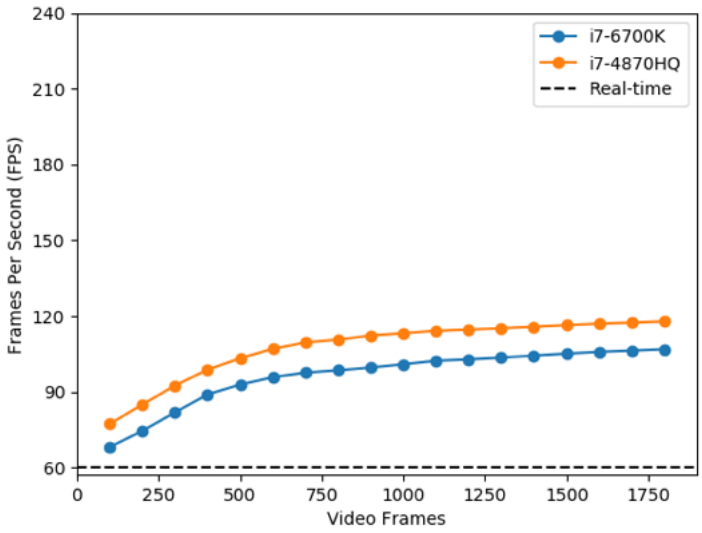


Figure A.6: Performance of the video processing system after Stage 5.

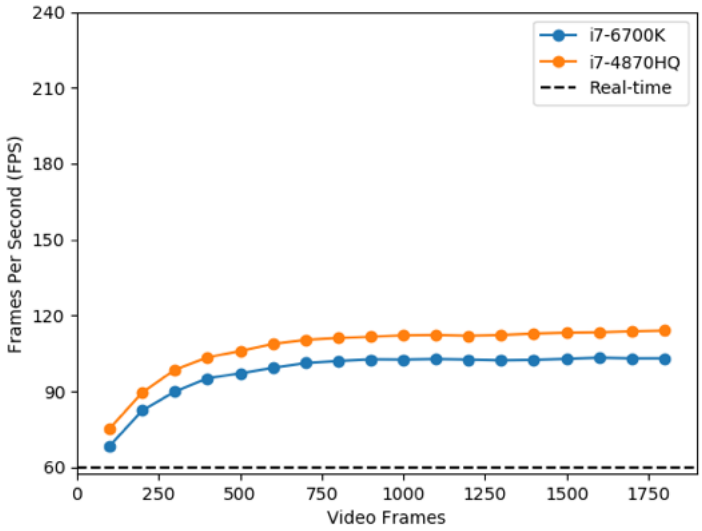


Figure A.7: Performance of the video processing system after Stage 6.

B

Program Outputs

```
Dec 3 12:26:43 2018: B*5011* in lane 1 with 114 km/h
Dec 3 12:26:43 2018:     None in lane 1 with  75 km/h
Dec 3 12:26:45 2018: BM*8*97 in lane 1 with 132 km/h
Dec 3 12:26:47 2018:  M*X0*8 in lane 1 with 108 km/h
Dec 3 12:26:49 2018: *A9506* in lane 1 with  85 km/h
Dec 3 12:26:51 2018: AV6*7*8 in lane 1 with 111 km/h
Dec 3 12:26:52 2018: H*5*096 in lane 1 with 111 km/h
Dec 3 12:26:53 2018: AM2*44* in lane 1 with  96 km/h
Dec 3 12:26:53 2018:  *RZ*48 in lane 1 with  94 km/h
Dec 3 12:26:55 2018: BN**541 in lane 1 with 112 km/h
Dec 3 12:26:54 2018:     None in lane 1 with  95 km/h
Dec 3 12:26:56 2018: BV*4*65 in lane 1 with  99 km/h
Dec 3 12:26:56 2018: A**2001 in lane 1 with  86 km/h
Dec 3 12:26:57 2018: AJ*210* in lane 1 with 108 km/h
Dec 3 12:26:59 2018: *L18*27 in lane 1 with  85 km/h
Dec 3 12:27:00 2018: ZC**634 in lane 1 with  75 km/h
Dec 3 12:27:01 2018: BV48*8* in lane 1 with  84 km/h

Found 17 vehicles in video.
Process ran in 92.87490464760691 frames per second.
```

Listing B.1: Output from running the video processor on a highway video. Two random characters in the license plated are replaced with * for privacy.

```
Dec 8 15:21:06 2018: KY*21*5 in lane 1 with 88 km/h
Dec 8 15:21:10 2018: VG*995* in lane 1 with 103 km/h
Dec 8 15:21:13 2018: FX*2*47 in lane 1 with 94 km/h
Dec 8 15:21:17 2018: T*37*88 in lane 1 with 97 km/h
Dec 8 15:21:20 2018: *L5533* in lane 1 with 84 km/h
Dec 8 15:21:24 2018: AA*4*72 in lane 2 with 108 km/h
Dec 8 15:21:27 2018: CT*83*7 in lane 1 with 104 km/h
Dec 8 15:21:31 2018: TL962** in lane 1 with 101 km/h
Dec 8 15:21:34 2018: RR7*57* in lane 1 with 95 km/h
Dec 8 15:21:38 2018: DM*61*7 in lane 2 with 112 km/h
Dec 8 15:21:41 2018: **12785 in lane 1 with 95 km/h
Dec 8 15:21:45 2018: XB*3*87 in lane 1 with 96 km/h
Dec 8 15:21:48 2018: *L2*668 in lane 1 with 86 km/h
Dec 8 15:21:52 2018: N*9*428 in lane 1 with 101 km/h
Dec 8 15:21:55 2018: QQ*8*72 in lane 1 with 98 km/h
Dec 8 15:21:59 2018: LS**214 in lane 1 with 89 km/h
Dec 8 15:22:02 2018: FO1*2*7 in lane 1 with 97 km/h
Dec 8 15:22:06 2018: TH*9*25 in lane 1 with 99 km/h
Dec 8 15:22:09 2018: ZI4*79* in lane 1 with 99 km/h
Dec 8 15:22:13 2018: FY*13*4 in lane 1 with 98 km/h
Dec 8 15:22:16 2018: *Y913*5 in lane 1 with 90 km/h
Dec 8 15:22:20 2018: *L99*25 in lane 1 with 86 km/h
Dec 8 15:22:23 2018: HJ*27*1 in lane 1 with 90 km/h
Dec 8 15:22:27 2018: YR*12*6 in lane 2 with 108 km/h
Dec 8 15:22:30 2018: Z*33*81 in lane 1 with 102 km/h
```

Listing B.2: Output from simulation with 1020 vehicles per hour and an average speed of 98 km/h. Two random characters in the license plates are replaced with * for privacy.

```
Dec 4 15:28:41 2018: Y*44*14 in lane 1 with  86 km/h(1)
Dec 4 15:28:43 2018: X*2*355 in lane 1 with 102 km/h(1)
Dec 4 15:28:44 2018: DZ*51*2 in lane 1 with 100 km/h(1)
Dec 4 15:28:46 2018: ZM6**33 in lane 1 with  95 km/h(1)
Dec 4 15:28:48 2018: SY*1*32 in lane 1 with  95 km/h(1)
Dec 4 15:28:49 2018: *X958*6 in lane 1 with  94 km/h(1)
Dec 4 15:28:51 2018: *U25*35 in lane 1 with  94 km/h(1)
Dec 4 15:28:53 2018: AL*5*89 in lane 1 with  90 km/h(1)
Dec 4 15:28:55 2018: ME78*8* in lane 1 with  95 km/h(1)
Dec 4 15:28:56 2018: PV*465* in lane 1 with  84 km/h(1)
Dec 4 15:28:58 2018: PZ45**5 in lane 1 with  97 km/h(1)
Dec 4 15:29:00 2018: *B923*3 in lane 1 with  98 km/h(1)
Dec 4 15:29:03 2018: L*41*41 in lane 1 with 101 km/h(1)
Dec 4 15:34:19 2018: XM**355 in lane 2 with 104 km/h(2)
Dec 4 15:34:34 2018: D*65*62 in lane 1 with  99 km/h(2)
Dec 4 15:34:53 2018: ZM6**33 in lane 1 with  94 km/h(2)
Dec 4 15:34:55 2018: L*4*241 in lane 2 with  97 km/h(2)
Dec 4 15:34:58 2018: SY*12*2 in lane 1 with  93 km/h(2)
Dec 4 15:35:00 2018: KX95**6 in lane 1 with  93 km/h(2)
Dec 4 15:35:01 2018: HU2*5*5 in lane 2 with  96 km/h(2)
Dec 4 15:35:06 2018: AB*23*3 in lane 2 with  94 km/h(2)
Dec 4 15:35:06 2018: P*4*455 in lane 2 with  96 km/h(2)
Dec 4 15:35:07 2018: ME*8*87 in lane 2 with  96 km/h(2)
Dec 4 15:35:17 2018: AL*52*9 in lane 1 with  87 km/h(2)
Dec 4 15:35:21 2018: YO*4*14 in lane 1 with  86 km/h(2)
Dec 4 15:35:57 2018: PV5*6*6 in lane 1 with  84 km/h(2)
```

Listing B.3: Output from simulator with two data collectors simulating 1530 vehicles per hour and an average speed of 98 km/h. Two random characters in the license plates are replaced with * for privacy.

```
Dec 5 12:46:09 2018: None in lane 1 with 96 km/h
Dec 5 12:46:10 2018: None in lane 1 with 84 km/h
Dec 5 12:46:13 2018: None in lane 1 with 111 km/h
Dec 5 12:46:17 2018: None in lane 1 with 90 km/h
Dec 5 12:46:23 2018: None in lane 1 with 89 km/h
Dec 5 12:46:27 2018: None in lane 1 with 95 km/h
Dec 5 12:46:29 2018: None in lane 1 with 92 km/h
Dec 5 12:46:32 2018: None in lane 1 with 36 km/h
Dec 5 12:46:32 2018: None in lane 1 with 76 km/h
Dec 5 12:46:34 2018: None in lane 1 with 144 km/h
Dec 5 12:46:35 2018: None in lane 1 with 68 km/h
Dec 5 12:46:33 2018: None in lane 1 with 62 km/h
Dec 5 12:46:36 2018: None in lane 1 with 82 km/h
Dec 5 12:46:37 2018: None in lane 1 with 73 km/h
Dec 5 12:46:39 2018: None in lane 1 with 92 km/h
Dec 5 12:46:43 2018: None in lane 1 with 72 km/h
Dec 5 12:46:45 2018: None in lane 1 with 57 km/h

Found 17 vehicles in video.
Process ran in 98.90228364346873 frames per second.
```

Listing B.4: Output from running the video processor on a rotated video.