Evaluating rain removal image processing solutions for fast and accurate object detection

A comparative study of two image processing algorithms in context of autonomous vehicles

TUGAY KÖZLÜOĞLU

LUKAS HENNICKS
Evaluating rain removal image processing solutions for fast and accurate object detection

A comparative study of two image processing algorithms in context of autonomous vehicles

TUGAY KÖYLÜOGLU & LUKAS HENNICKS
Abstract

Autonomous vehicles are an important topic in modern day research, both for the private and public sector. One of the reasons why self-driving cars have not yet reached consumer market is because of levels of uncertainty. This is often tackled with multiple sensors of different kinds which helps gaining robustness in the vehicle’s system. Radars, lidars and cameras are often the sensors used and the expenses can rise up quickly, which is not always feasible for different markets. This could be addressed with using fewer, but more robust sensors for visualization. This thesis addresses the issue of one particular failure mode for camera sensors, which is reduced view range affected by rainy weather. Kalman filter and discrete wavelet transform with bilateral filtering are evaluated as rain removal algorithms and tested with the state-of-the-art object detection algorithm, You Only Look Once (YOLOv3). Filtered videos in daylight and evening light were tested with YOLOv3 and results show that the accuracy is not improved enough to be worth implementing in autonomous vehicles. With the graphics card available for this thesis YOLOv3 is not fast enough for a vehicle to stop in time when driving in 110km/h and an obstacle appears 80m ahead, however an Nvidia Titan X is assumed to be fast enough. There is potential within the research area and this thesis suggests that other object detection methods are evaluated as future work.

Keywords: object detection, failure modes, autonomous vehicles
Sammanfattning


Nyckelord: objektigenkänning, felmodell, autonoma fordon
Acknowledgements

We would like to thank De-Jiu Chen for supporting us through this thesis. Thank you to Tor Ericson, Björn Eriksson and Adrian Nadi for giving us the opportunity to do our thesis at ÅF in Solna. We would also like to thank Fredrik Asplund for feedback regarding our research questions. And finally we would like to thank our opponent André Säll for feedback after our final presentation.

Tugay Köyluoglu & Lukas Hennicks
Stockholm, May, 2019
Contents

List of Figures v
List of Tables vi
Nomenclature vii

1 Introduction 1
   1.1 Background ........................................ 1
   1.2 Problem description ............................... 2
   1.3 Purpose and definitions .......................... 3
   1.4 Scope ............................................. 4
   1.5 Methodology ....................................... 5
       1.5.1 Ethics ....................................... 6

2 Theory 7
   2.1 Failure Modes ....................................... 7
       2.1.1 Failure Modes for a Camera .................. 7
   2.2 Autonomous Vehicle Dynamics ...................... 8
       2.2.1 Braking Dynamics ............................. 8
       2.2.2 Effects of Latency ......................... 10
   2.3 Vehicle Localization and Mapping .................. 11
   2.4 Rain Dynamics in Videos ........................... 12
   2.5 Image Processing .................................. 13
       2.5.1 Filter ....................................... 13
       2.5.2 Gaussian Blur ............................... 13
       2.5.3 Convolution ................................. 13
       2.5.4 Bilateral Filtering .......................... 14
       2.5.5 Kalman Filter ............................... 17
       2.5.6 Rain Removal Using Kalman Filter ............ 19
       2.5.7 Wavelet Transform ........................... 19
## CONTENTS

2.5.8 Rain Removal Using Discrete Wavelet Transform and Bilateral Filtering ............................................. 21
2.6 Object Detection ................................................................................................................................. 23
  2.6.1 Artificial Neural Networks ........................................................................................................... 24
  2.6.2 Convolutional Neural Networks ................................................................................................. 25
  2.6.3 You Only Look Once ................................................................................................................. 26
2.7 GPU-Accelerated Computing ........................................................................................................... 28
2.8 Related Work ................................................................................................................................... 29
  2.8.1 Conclusion ................................................................................................................................. 30

3 Implementation ................................................................................................................................. 31
  3.1 Vehicle Dynamics Simulation ......................................................................................................... 31
  3.2 OpenCV ........................................................................................................................................... 32
  3.3 YOLOv3 .......................................................................................................................................... 33
  3.4 Rain Removal Using Kalman Filter ............................................................................................... 33
  3.5 Rain Removal Using Discrete Wavelet Transform and Bilateral Filtering ....................................... 34
  3.6 Testing ............................................................................................................................................ 35
    3.6.1 Combining Rain Removal with the Object Detection ................................................................. 36
    3.6.2 Resolution and Downscaling ..................................................................................................... 37
    3.6.3 Testing the Rain Removal .......................................................................................................... 38
    3.6.4 Measuring YOLOv3 Accuracy ................................................................................................. 38
    3.6.5 Measuring Speed ....................................................................................................................... 39

4 Results ............................................................................................................................................... 41
  4.1 Rain Removal Performance ............................................................................................................. 41
  4.2 Accuracy .......................................................................................................................................... 42
    4.2.1 Intersection over Union ............................................................................................................ 42
    4.2.2 Confidence Level ....................................................................................................................... 45
  4.3 Impact of Resolution ....................................................................................................................... 49

5 Evaluation ......................................................................................................................................... 52
  5.1 Reliability of Ground Truth ............................................................................................................. 52
  5.2 Night Time Rain Removal ................................................................................................................. 52
  5.3 Evaluation of Snow Removal ............................................................................................................ 54
  5.4 Object Detection Latency .................................................................................................................. 55
  5.5 Hardware Requirements .................................................................................................................... 55
    5.5.1 Vehicle Scenarios ...................................................................................................................... 55
    5.5.2 Computational Requirements ................................................................................................. 57
# List of Figures

2.1 Figures and results from Nayar and Garg [27].  
2.2 Before Gaussian blur on the left and after on the right. For the filter, a kernel size of 10 pixels was used [29].  
2.4 Before bilateral filtering on the left and after bilateral filtering on the right. For the filter, $\sigma_r = 2$ and $\sigma_s = 200$ are used [29].  
2.5 Gaussian distribution with mean $\mu$ and standard deviation $\sigma$ [35].  
2.6 Two common wavelets. A: Haar Wavelet, B: db4 wavelet [37].  
2.7 Filter bank for 3-Level DWT decomposition [38].  
2.8 Chart of the rain map extraction process [8].  
2.9 A single neuron [43].  
2.10 Connected Neurons with weights [43].  
2.11 Structure of a simple CNN [45].  
2.12 Example of a YOLO grid [12].  
2.13 YOLO grid with predicted bounding boxes. Thicker border indicate stronger confidence [12].  
2.14 The final resulting bounding boxes [12].  
3.1 Simulink model for ABS braking.  
3.2 Comparison of the video with added rain with the original video [60].  
3.3 Frame of video in daylight [60] with YOLOv3 running through it.  
4.1 A: No rain removal, B: Kalman Filter, C: DWT&BF.  
4.2 IoU in percentage for DWT & BF, KF and added rain with ground truth in 1080p. Video is a traffic video in daylight.  
4.3 Average IoU in percentage for KF, added rain and DWT & BF with ground truth in different pixel resolutions.
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4</td>
<td>Confidence level difference in percentage points for DWT &amp; BF and KF with ground truth and rain with ground truth in 1080p.</td>
<td>46</td>
</tr>
<tr>
<td>4.5</td>
<td>Average confidence level difference in percentage points for KF, DWT &amp; BF and added rain with ground truth in different pixel resolutions.</td>
<td>47</td>
</tr>
<tr>
<td>4.6</td>
<td>Average amounts of detection in 100% confidence level for KF, DWT &amp; BF, ground truth and added rain in different pixel resolutions.</td>
<td>48</td>
</tr>
<tr>
<td>5.1</td>
<td>Night time frame without any filtering [63].</td>
<td>53</td>
</tr>
<tr>
<td>5.2</td>
<td>DWT &amp; BF processed frame at night time. The result with image inpainting can be seen on the left and the rain map generated can be seen on the right [63].</td>
<td>53</td>
</tr>
<tr>
<td>5.3</td>
<td>KF processed frame at night time [63].</td>
<td>53</td>
</tr>
<tr>
<td>5.4</td>
<td>Demonstration of DWT edge mapping only detecting the edges of snowflakes. The original is to the left and the map is on the right.</td>
<td>54</td>
</tr>
<tr>
<td>5.5</td>
<td>Stopping distances with $v_0$ as starting velocity. X-axis shows time to full stop and Y-axis shows braking distance in meters.</td>
<td>56</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Parameters for a wheel of a car. .............................. 9

3.1 Parameters of Tesla Model S P100D for simulation of stopping distances. ............................... 32

3.2 Table over tested video resolutions. .......................... 37

4.1 Table of the average pixel intensity difference compared to the ground truth. ............................... 41

4.2 Table over results for daytime traffic video with KF and YOLOv3. 49

4.3 Table over results for daytime traffic video with DWT & BF and YOLOv3. ............................... 49

4.4 Table over results for traffic video during evening lighting condition with KF and YOLOv3. 50

4.5 Table over results for traffic video during evening lighting condition with DWT & BF and YOLOv3. 50

4.6 Table over results for daytime traffic video with KF and Tiny YOLOv3. ............................... 50

4.7 Table over results for daytime traffic video with DWT & BF and Tiny YOLOv3. 50

4.8 Latency of image inpainting. ............................... 51

5.1 Table over results for daytime traffic video with KF and YOLOv3. 57
## Nomenclature

### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>Anti-lock Braking System</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma-Separated Values</td>
</tr>
<tr>
<td>CUDA</td>
<td>Compute Unified Device Architecture</td>
</tr>
<tr>
<td>DWT &amp; BF</td>
<td>Discrete Wavelet Transform and Bilateral Filtering</td>
</tr>
<tr>
<td>FPS</td>
<td>frames per second</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>IoU</td>
<td>Intersection over Union</td>
</tr>
<tr>
<td>KF</td>
<td>Kalman Filter</td>
</tr>
<tr>
<td>OD</td>
<td>Object Detection</td>
</tr>
<tr>
<td>OpenCV</td>
<td>Open source Computer Vision</td>
</tr>
<tr>
<td>PP</td>
<td>percentage points</td>
</tr>
<tr>
<td>SIMD</td>
<td>Single Instruction Multiple Data</td>
</tr>
<tr>
<td>SLAM</td>
<td>Simultaneous Localization and Mapping</td>
</tr>
<tr>
<td>YOLOv3</td>
<td>You Only Look Once v3</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This chapter introduces the background of the thesis. It presents the problem and purpose of this thesis and describe the limitations and method of work.

1.1 Background

This thesis is conducted at ÅF AB, an engineering and design company in Stockholm, Sweden. The company creates sustainable and digital solutions and specializes within the fields of energy, industry and infrastructure. ÅF AB is based in Europe with clients all over the world [1].

The company provide their customers both with products and research in several topics. One area of expertise is regarding the field of autonomous vehicles where the company provide consultation to different companies [2].

The industry of autonomous vehicles is growing fast and has become a large topic in modern time research. In an era where the technology industry is growing rapidly, improvements of computing power opens up new opportunities for autonomous vehicles. With the advancements in computational power, machine learning and artificial intelligence advanced systems has been able to be implemented in vehicles, with sensors such as cameras, radars and lidars, to act as the vehicle’s eyes [3].

Research on autonomous vehicles is ongoing and consumer grade autonomous vehicles have not yet reached the consumer market. One of the reasons why is because of the required robustness. Since one error could be fatal it is crucial to reduce uncertainties [4]. One challenge with robustness is failure modes
for the different sensors. Failure modes are the different ways in which a sensor can fail. This is commonly solved with multiple redundant sensors. With redundancy, these kinds of systems can, as of today, operate at the level of a human driver [5] [4]. Implementing redundancy by fusing multiple sensors can quickly become expensive and unfeasible to many consumers [6]. A problematic failure mode for camera based sensors is when the view range is reduced due to weather conditions such as rain and snow [7].

1.2 Problem description

To solve the issues of failure modes, acquiring redundancy by using multiple sensors could be a solution. However, with cost constraints, it could be more feasible to only use a single camera as a sensor for object detection in autonomous vehicles. The failure modes could be counteracted with more cost efficient solutions. Furthermore, even if this problem was easier to solve with redundant sensors, the research of rain/snow removal can have many applications and the research can apply to many fields [5].

Rain is affecting visual systems such as object tracking and classification since it causes complex spatio-temporal intensity fluctuations in videos [8]. Cameras in vehicles could be placed in different positions, for example, the camera could be positioned inside the car or outside the car. The lens could be protected from splashes or not protected at all, creating different types of distortion from the rain. Since rain is affecting the image quality of a camera, an autonomous vehicle should hypothetically navigate better in rainy weather if the rain was to be removed from the video data.

Autonomous vehicles needs to compute large amounts of data in short amounts of time in order to navigate through busy areas by itself. Many objects pass by fast and it is critical for the system to detect the objects in order to navigate. If it is supposed to replace a human driver, the autonomous vehicle’s decision making must be faster than a human’s reflexes and still maintain good accuracy.

One solution for dealing with rain in videos is by removing them with filters. For autonomous vehicles, the filters should be good enough to remove rain without compromising too much detail and fast enough to not become a bottleneck of the object detection algorithm. The resolution of the camera will also affect the speed and accuracy of the filters and will therefore be a parameter to optimize in order to gain optimal accuracy and speed. As of to-
day, no sources indicate any usage of rain removal algorithms in autonomous vehicles, navigation in poor weather conditions is however being researched. The wide research area includes not only camera but also other sensor types such as lidar and ultrasonic sensors. A small selection of articles regarding this research area are the following three: [9] [10] [11].

1.3 Purpose and definitions

The purpose of this thesis was to evaluate two proposed methods of removing rain along with an object detection algorithm. The methods of removing rain were both focused on removing droplet like features which could cause disturbances. In this thesis, the pre-trained YOLOv3 object classification algorithm was used [12]. The first method of removing rain was using the discrete wavelet transform and bilateral filtering [8] jointly in order to detect and then filter out the raindrops while the second method used a Kalman filter [13]. These methods were tested in conjunction with the object detection algorithm in order to measure how the accuracy and the latency of the object detection varied between the different methods and when no rain removal was used. Another topic that the thesis investigated was how this accuracy varied if the resolution was lowered. Downscaling the resolution can lead to much lower latency so it was valuable to find how low resolution could be acceptable. This can be formulated as the following research questions:

In the context of a camera used as an object classifier to help an autonomous vehicle brake in a critical situation, such as those mentioned earlier. For the view range reducing failure mode caused by rain:

**RQ1:** How does the two solutions listed above compare to each other, and to the object classifier without rain removal, in terms of classifier accuracy and speed?

**RQ2:** Given the accuracy of the object classifier along with the rain removal algorithms, what resolution of the camera is optimal in terms of speed while maintaining the accuracy?

Seen from a wider perspective, the results can touch upon many challenges regarding navigation of an autonomous vehicle in crowded areas and in heavy rain. Thus the aspects of how an end to end product could be delivered can be explored by discussing questions as:
• What hardware is required in the vehicle to allow a robust object detection system to perform well?

• How fast does the object detection have to be for the vehicle to be safe?

• What is optimal regarding the resolution/accuracy trade-off?

1.4 Scope

This thesis had the following delimitations:

• The rain removal algorithms used were based on removing droplets of rain in relatively close proximity (around 1-100m), not counteracting the possible reduced view range rain can cause at further distances where terrain can seem blurry.

• Other methods such as deep neural networks that are trained to see objects through rain are not investigated in this thesis [14] [15].

• Tests was done on an Nvidia Quadro P1000 [16].

• Settings such as aperture, shutter speed, ISO and choice of camera and lens were not evaluated.

• The object detection algorithm used was pre-trained and no changes were done to the algorithm or the trained network itself.

• This thesis only investigated the reduced view range failure mode for a camera. It did not investigate blurred view caused by rain.

• Algorithms and methods used for autonomous driving, such as SLAM and object detection, were used and briefly explained in the thesis but was not further investigated than necessary.

This thesis did not discuss a design of a new method. It only investigated already existing methods and evaluated them in the context of an autonomous vehicle system.
1.5 Methodology

In order to solve the tasks set for this thesis, a frame of reference was conducted by collecting sources about theories and methods on related research. The choice of rain filtering methods as well as object detection method were derived from literature studies. These methods could later be evaluated in the context of autonomous vehicles in order to reduce the risk of collision. To pose timing requirements of the rain removal and object detection pipeline, simulations of stopping distances for a vehicle were done. By calculating a maximum allowed latency it could be used as a reference when reasoning about the speed of the algorithms. Hardware requirements was defined based on the timing requirements posed from the stopping distance simulations.

YOLOv3 was initially implemented in order to evaluate its baseline performance. Following the implementation of YOLOv3, a first implementation of the rain removal methods was planned. After this, videos could be rendered out to run through the YOLOv3 algorithm, allowing testing and evaluation.

The main metric used for evaluating the accuracy was the Intersection over Union. This was chosen since it is one of the most reliable ways of measuring accuracy for bounding boxes. While other metrics may be common in other kinds of algorithms, in object detection, the position and size of the bounding boxes is what matters. Several sources indicate that IoU is a standard when it comes to object detection accuracy and in the YOLOv3 report it is used as a metric for evaluating performance [12] [17].

Results were planned to be validated by comparing the object detection results. The comparisons were done between footage with rain, a ground truth without rain and both rain removal methods. Different video resolutions were tested in order to discuss the trade-offs in regards to accuracy and latency. Results were also planned to be discussed in the context of an entire autonomous vehicle.

Since it was a risk that the algorithms chosen for removing rain would not work and was a single point of failure, basic implementations were made early on.
1.5.1 Ethics

Ethics is a large topic when discussing autonomous vehicles. One common discussion is about decisions when risking an accident. For example, if there are five pedestrians on the road and one on the sidewalk and the vehicle doesn’t have the time to break, should it hit one person or five? The issues of choosing who to prioritize have been a philosophical discussion and is in today’s industry a general concern [18].

In this thesis these kinds of decisions are not an issue since improving the system is the main objective. However, regarding the accuracy/latency trade-off there were some concerns. For example, an ethical consideration was if the technology investigated were used instead of redundant sensing and therefore could cause more accidents. This thesis does however not state what system someone should use but does mainly investigate the performance of already existing methods. If someone chooses to use these methods in autonomous vehicles, these ethical considerations are in their hands.

Another ethical aspect of autonomous vehicles is the handling of data for testing and improving the system. Machine learning demands data and in the context of vehicles are training data of humans and cars used. A possible ethical issue could be how this data is used and if any consent exists [18]. In this thesis the test data of pedestrians and cars was gathered from license-free videos. It is not known for sure if the people involved in these videos have given their consent or not. The main objective of this thesis is to improve liability and confidence of an object detection system for autonomous vehicles. The trained network of the object detection system that was being used for this thesis used data containing lots of people. Evaluating methods to improve vision in rainy weather could be seen as an important ethical question just as using data of people. Since ethics is a philosophical topic is it difficult to conclude what is the right answer and what is wrong.

One ethical consideration is the bias of investigating the data resulted by the work of the authors of this thesis themselves. Risks of an objective analysis being partly subjective might exist and the solution for this thesis was to consult a third party for confirming the results and conclusions.
Chapter 2

Theory

2.1 Failure Modes

A failure mode is a possible way that a sensor could yield invalid results. Different types of sensors can have different failure modes and different amounts of them as well.

2.1.1 Failure Modes for a Camera

Cameras have improved a lot over the last century and often perform well in good weather conditions, however could the performance suffer in certain weather conditions. Heavy rain, snow, fog, glare of the sun, sand and occlusion are a few weather conditions that could affect an imaging sensor. The quality compromising failure modes, caused by weather for an image sensor are the following [7]:

- Blocked view: Occlusion or blocking the lens from example ice or issues with the sensor.
- Blurred view: Dirt or soil on the lens, cracks on the lens or other damages that cause optical faults.
- Reduced view range: Poor weather such as fog, heavy rain or snowfall that reduces the range of view.
- Misaligned view: Faulty mechanical housing for the camera or poor mounting.

Some are easier to solve than others for example dirt or soiled lenses can be cleaned as well as blockage or if the lens is broken it can be replaced. However
it is much more difficult to avoid the reduced view range for an imaging sensor. For autonomous vehicles it is important to have a long view since the cameras job is to detect and classify objects such as traffic signs, pedestrians and other detailed information that the lidar and radar cannot.

2.2 Autonomous Vehicle Dynamics

The complexity of vehicle dynamics extends far beyond a thesis in mechatronics. In the following subsection basic theory of braking dynamics will be provided in order to complement the subsection 2.2.2 Effects of Latency. This background will be discussed further on in Chapter 3 and 5.

2.2.1 Braking Dynamics

In this subsection a mathematical model of braking dynamics will be provided. These equations can be further used for simulations in order to estimate braking distances. This model will be fairly simple, but with the essentials still included it is complex enough for this thesis. The model will be nonlinear and compute braking dynamics with respect to one-wheel rotational dynamics. Newton’s law will be used for the wheel and vehicle dynamics. The angular motion (2.1a) and vehicle motion (2.1b) are shown in the following equations [19]:

\[
\begin{align*}
\dot{\omega}_w &= \frac{T_e - T_b - R_w F_t - R_w F_w}{J_w} \quad (2.1a) \\
\dot{v} &= \frac{N_w F_t - F_v}{M_v} \quad (2.1b)
\end{align*}
\]

The drag force \( F_v \) [20] and viscous friction \( F_w \) [21] equations are provided below:

\[
\begin{align*}
F_v &= \frac{C_d A \rho v^2}{2} \quad (2.2a) \\
F_w &= \mu_s N_v \quad (2.2b)
\end{align*}
\]

The definition of the traction/braking force \( F_t \) is shown in Equation 2.3 and definitions of all other parameters are shown in Table 2.1:
$F_t = \mu(\lambda)N_v$ \quad (2.3)

where $\mu(\lambda)$ is the adhesion coefficient which depends on the wheel slip $\lambda$. This nominal function is shown below in Equation 2.4:

$$\mu(\lambda) = \frac{2\mu_p\lambda_p\lambda}{\lambda_p^2 + \lambda^2} \quad (2.4)$$

where $\mu_p$ and $\lambda_p$ are the peak values which depend on various road conditions. This function can be further explained with the following equations [19]:

$$\omega_v = \frac{v}{R_w} \quad (2.5a)$$

$$\omega = \max(\omega_w, \omega_v) \quad (2.5b)$$

$$\lambda = \frac{\omega_w - \omega_v}{\omega}, \omega \neq 0 \quad (2.5c)$$

The vehicle angular velocity and angular wheel velocity can be set as states in order to simulate the vehicle motion and angular wheel motion. Equations are shown below:

$$x_1 = \frac{v}{R_w} \quad (2.6a)$$

$$x_2 = \omega_w \quad (2.6b)$$

$$x = \max(x_2, x_1) \quad (2.6c)$$
where the motion states can be described as following:

\[
\dot{x}_1 = -f_1(x_1) + \frac{N_v N_w}{M_v R_w} \mu(\lambda) \\
\dot{x}_2 = -f_2(x_2) - \frac{R_w N_v}{J_w} \mu(\lambda) + \frac{T_e - T_b}{J_w}
\]

(2.7a) (2.7b)

where \( f_1(x_1) \) and \( f_2(x_2) \) are nonlinear equations and the wheel slip \( \lambda \) is:

\[
f_1(x_1) = \frac{F_v(R_w x_1)}{M_v R_w} \\
f_2(x_2) = \frac{F_w(x_2)}{J_w} \\
\lambda = \frac{x_2 - x_1}{x}
\]

(2.8a) (2.8b) (2.8c)

### 2.2.2 Effects of Latency

The reaction time of a human driver can differ between different drivers and also depend on various variables such as the level of tiredness, concentration, experience, etc. One of the arguments for an autonomous vehicle is the reduced reaction times because of less latency and no human errors. In the context of braking situations the reaction time is an important factor and can be the difference between life or death. For example when driving in 90km/h, which can be translated to 25m/s, the car has traveled 50m if the reaction time was 2s before initialized braking has begun. The effects of latency in braking situations can therefore be hazardous. The braking process can be divided into three phases: the delay from reaction time, activation to full deceleration and finally maximum braking force where this subsection focuses on the delay/latency until actual braking.

Upper and lower bounds for estimated reaction times (delay and physical braking activation) can differ from 1.3-3.5s for a human and 0.011-0.2s for a computer [22]. The distance traveled during the reaction time when driving at 100km/h could be as long as 36.1-97.3m for a human driver and 0.306-5.56m for a machine with the upper and lower bounds provided from Browne et al. [22]. The differences in distances are large and can still be improved with faster processors and algorithms while human reaction times are limited.
2.3 Vehicle Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is a concept commonly used in robotics and autonomous vehicles. As the name suggests, it is used to simultaneously localize the current position of for example a vehicle, while also mapping the surrounding environment. SLAM algorithms are usually based on multiple inputs of sensor data. [23]. Extended Kalman filter-SLAM (EKF-SLAM) and Rao-Blackwellized particle filter (Fast-SLAM) are two common SLAM algorithms. Some common sensor types used for SLAM are lidar, radar, GPS and cameras. The sensor data from cameras are commonly used in order to detect landmarks and visual odometry, making it important that the image quality is as good and noise free as possible.

Placement of camera sensors is as important as having good image quality. Poor placement could affect the field of view and leave unsupervised areas. Cameras could be placed in many different ways and research suggesting a specific positioning was not found. Different car manufacturers place their cameras in different places and use different kinds of focal lengths with their lenses. Cameras could be placed in front of vehicles, at the sides, behind the wind shield, in the rear or on the roof of the vehicle. Wide angle lenses would solve problems with poor field of view but also compromise detail because of lens distortion. Using a large amount of cameras could also be a solution, however would that be more expensive and more complex in computation.

With the increase of computational power from CPU’s and GPU’s it has now been possible to deploy machine learning and deep learning algorithms to vehicles in order to increase their levels of autonomy. Algorithms within regression, pattern recognition, clustering and decision matrices are sections of machine learning that could be used within autonomous driving. Deep learning, a form of artificial neural networks (which is explained in Subsection 2.6.1) but with multiple hidden layers, is the most commonly used in self driving cars since the levels of complexity is able to increase tremendously. Neural networks have been endorsed by Tesla and Waymo and is able to detect objects, make decisions and much more[24].

Improved pattern recognition and object detection with machine learning and deep learning would make data from camera sensors more reliable and useful for SLAM algorithms. Filtering the data from rain during rainy weather conditions would further improve SLAM.
2.4 Rain Dynamics in Videos

Rain is a dynamic weather condition with particles of sizes between 0.1 - 10 mm in diameter and compared to steady weather such as fog with particles in sizes 1 - 10 µm, rain particles are more visible to cameras [25]. Since cameras capture video in multiple frames each second, rain is introduced as sharp intensity patterns that fluctuate between frames, resulting in compromised video quality [26][27]. Each water droplet acts as a transparent sphere that refracts and reflects different light intensities from the environment to the camera sensor and with finite exposure time will each drop create motion blur in the image. The effect of these fluctuations is partly dependent on the background. How the images are affected by the rain differs with camera settings. However, these camera parameters then affect the video characteristics [27] such as exposure, noise and frame rate. Velocity of rain is also a parameter that together with background and camera parameters affect the complex spatio-temporal intensity fluctuation [27].

According to K. Garg and S. K. Nayar experiments has been done where pixel intensities created by rain with different backgrounds has been measured [27]. The results show that the intensities peaks at up to approximately the same values of 200, out of 255 being the maximum 8-bit intensity. The intensity of the rain particles in their experiments were always higher than the backgrounds. Their experimental results can be shown in Figure 2.1a, and in Figure 2.1b.

(a) Pixel intensities with different background lightings. Brightness of the drops are almost the same and always brighter than the background.

(b) Illustration of pixel intensity frame by frame.

Figure 2.1: Figures and results from Nayar and Garg [27].
2.5 Image Processing

2.5.1 Filter

Filtering is a process which can remove unwanted features from a signal. These could be anything from noise or disturbances to bright areas or sharp edges. While filtering is used on many different types of signals, such as audio signals or control signals, this report will focus on the use of filters in image processing.

2.5.2 Gaussian Blur

One of the most common ways an image is filtered is with a Gaussian low pass-filter. It is often used in order to make noise less visible by blurring the entire image. The Gaussian low pass-filter works by calculating new values for each pixel based on a weighted average of the other pixels of the image. In a Gaussian low pass-filter the pixels are weighted based on a normal distribution, for the mathematical definition of a two dimensional Gaussian blur, see Equation 2.9. [28]

\[ g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \]  

In the equation \( \sigma \) is the standard deviation of the Gaussian distribution and \( x, y \) are the horizontal and vertical distances from the origin (for blurring an image, this is done with each pixel as an origin). The amount of blur caused depends on how wide the standard deviation is set. The wider the standard deviation is set, more pixels are weighed in, resulting in a more generalized average and more blur. In practice this variant of the Gaussian blur is computationally expensive. For each pixel an average has to be calculated of all pixels in the entire image, resulting in a computational complexity of \( \mathcal{O}(n^2) \), where \( n \) depends on the number of pixels in the image. To overcome this, Gaussian blur implementations often convolve the image with a kernel with its weights set as a radial normal distribution. In this case, the amount of necessary calculations are greatly reduced since only the size of the kernel is calculated for each pixel. An example of Gaussian blurring can be seen in Figure 2.2. [28]

2.5.3 Convolution

Convolution is a type of operation which can be applied to many different kinds of signals. Examples of fields where convolution is useful are audio processing, control theory and image processing. This part will focus on convolution
in the context of image processing.

Intuitively explained, convolution uses a small square matrix, often referred to as a kernel, which is slid over each pixel in an image, allowing each new pixel that is calculated to include surrounding pixels in its calculations. The values of the kernel can be set as desired which in turn causes the surrounding pixels to be weighed differently. This can in turn cause drastically different outputs. Mathematically, the operation of convolution can be expressed as Equation 2.10, where \( g(x, y) \) is the convolved output, \( f(x, y) \) is the original image and \( w \) is the filter kernel. The kernel size is \( 2a \times 2b \) making each element of the kernel considered with the two sums which range from \(-a \leq s \leq a\) and \(-b \leq t \leq b\).

\[
g(x, y) = (\omega * f)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x - s, y - t)
\]

Convolution is very common in image processing and is often used for filtering images. Examples of use-cases are in Sobel edge detection or different kinds of blurring (box blur, Gaussian blur, etc). [30]

### 2.5.4 Bilateral Filtering

While a Gaussian blur can be effective at removing noise, one of its major downfalls is how edges also get blurred and thus become harder to detect. This
is what a bilateral filter excels at, blurring out noise while preserving edges. [31]

The major difference between the two is how a bilateral filter has a term $G_{\sigma_r}$ that is called the range weight, which recognizes edges and modifies the shape of the Gaussian kernel to not average across the edges. It does however also include the $G_{\sigma_s}$ term which is called the spatial weight. The spatial weight is the same as the Gaussian kernel. An illustration of the spatial weight and range weight can be seen in Figures 2.3a-2.3c which are all taken from A Gentle Introduction to Bilateral Filtering and its Applications [32]. Note how all figures are just based on the kernel in one position. One of the core concepts of bilateral filtering is that the kernel is calculated separately for each position. [32]
(a) To the left is the spatial kernel weights and to the right is the range kernel weights for an image edge [32].

(b) Combined kernels [32].

(c) The left image is the original data and the right image is how it looks after applying the kernel from Figure 2.3b [32].

The full equation for the bilateral filter can be seen in Equation 2.11, where $I_p$ is the intensity of a pixel $p$, $W_p$ is a normalization term, $p$ is the position of the pixel in the center of the kernel, $I_q$ is the pixel intensity of pixel $q$ where $q$ is the position of the other pixels of the kernel, $S$ is the set of pixels in the
In bilateral filtering there are therefore two parameters that can be tweaked, $\sigma_s$ and $\sigma_r$. $\sigma_s$, the spatial parameter, smooths out larger features if increased. $\sigma_r$, the range parameter, causes the bilateral filter to become closer to a Gaussian blur if increased since it causes the kernel to be affected less by sharp edges. A before and after bilateral filtering can be seen in Figure 2.4. [32]

![Before and after bilateral filtering](image)

**Figure 2.4:** Before bilateral filtering on the left and after bilateral filtering on the right. For the filter, $\sigma_r = 2$ and $\sigma_s = 200$ are used [29].

### 2.5.5 Kalman Filter

Kalman filtering (KF) is an algorithm used for combining prior data with measurements in order to optimize a posterior estimation for filtering and prediction systems. This filter is a Gaussian filter. The Kalman filter works with linear dynamics however with the extended Kalman filter it is possible to work with nonlinear dynamics. [33] KF is suitable for sensor fusion where multiple measurements and prior estimations are combined for a posterior estimation with a narrower Gaussian distribution and accurate mean value. [34] An illustration of the Gaussian distribution can be seen in Figure 2.5 and the equation can be seen in Equation 2.12 [33] where $\sigma^2$ is the variance and $\mu$ is the mean value.

\[
p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi \sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}
\]  

(2.12)
Kalman filters are stateful and makes use of Markov properties. This means that the current state is only affected by the previous state and action, not a sequence of events. At the current state in time stamp $t$, the belief is characterized by the mean $\mu_t$ and covariance $\Sigma_t$. Kalman filters can be divided into two fundamental steps: predictions and updates. The prediction step of the filter are the following equations:

$$
\hat{\mu}_t = A_t \mu_{t-1} + B_t u_t + \epsilon_t \\
\epsilon_t \sim \mathcal{N}(0,R_t) \\
\hat{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t
$$  \hspace{1cm} (2.13a, 2.13b, 2.13c)

where $\hat{\mu}_t$ and $\hat{\Sigma}_t$ are the predicted mean and covariance states, $\mu_{t-1}$ and $\Sigma_{t-1}$ are the current mean and covariance states, $u_t$ is the input state, $\epsilon_t$ is Gaussian process noise with zero mean and covariance $R_t$ and $A_t$ and $B_t$ are transition matrices from current state to the predicted state. A measurement and Kalman gain step is needed in order to update the predictions. These equations are explained below:

$$
z_t = H_t \mu_t + \delta_t \\
\delta_t \sim \mathcal{N}(0,Q_t) \\
K_t = \hat{\Sigma}_t H_t^T (H_t \hat{\Sigma}_t H_t^T + Q_t)^{-1}
$$  \hspace{1cm} (2.14a, 2.14b, 2.14c)

where $H_t$ is the measurement transition matrix. $\delta_t$ is the measurement noise with zero mean and measurement covariance $Q_t$. $z_t$ is the measurement of the mean state and $K_t$ is the Kalman gain. The final equations for updating the state are the following:

$$
\mu_t = \hat{\mu}_t + K_t(z_t - H_t \hat{\mu}_t) \\
\Sigma_t = (I - K_t H_t) \hat{\Sigma}_t
$$  \hspace{1cm} (2.15a, 2.15b)
where $I$ is the identity matrix. [33]

According to Thrun et al. [33] Kalman Filters have a Gaussian posterior if the following three conditions are held:

- The probability of the predicted state $p(\hat{\mu}_t|u_t, \mu_{t-1})$ must be a linear function with added Gaussian noise.
- The probability of the measurement $p(z_t|\mu_t)$ must be a linear function with added Gaussian noise.
- The initial belief $p(\mu_0)$ must be Gaussian.

### 2.5.6 Rain Removal Using Kalman Filter

Kalman filters are well suited for estimating the following state based on the information from the current state. Park and Lee [13] discuss their implementation of KF for removing rain on videos and their method is what was tested in this thesis.

The method works as such that the current state is the current frame in the video and the following state is the next frame. In order for this algorithm to work it is needed to be one frame behind in the process and with a set threshold find the pixels with temporally raised intensities. With the pixels located, the next frame with higher pixel intensities could be replaced with the new predicted and later on corrected frame where the intensities are estimated to a lower value for each additive colour red, green and blue. Important parameters in the Kalman filter are the process noise covariance $Q_t$ and the measurement noise covariance $R_t$. Setting a larger value for one of the covariances will make the filter more biased towards that particular side and since rain is seen as noise in the Kalman filter a larger $Q_t$ value could favour there being less rain in the frame.

### 2.5.7 Wavelet Transform

The wavelet transform is a way of expressing a signal as a collection of smaller wavelets. This is in contrary to common methods such as describing the signal as a collection of frequencies (Fourier transform) or a signal level at each sample in time. In the example of a Fourier transform, there is no concept of where in time a specific frequency is occurring. In describing the signal as a
sampled value at each sample time interval, there is on the other hand no information regarding the frequency. This is where the wavelet transform can act as a compromise, supplying both temporal properties and frequency information.

Intuitively, a wavelet can be seen as a waveform that, instead of having a constant amplitude, fades in and out at a specific point in time. The core concept is to rebuild the original signal with an infinite amount of small wavelets. Some sample wavelets can be seen in Figure 2.6 [36]. Mathematically, the wavelet transform can be described according to Equation 2.16

$$F(s, \tau) = \int_{-\infty}^{\infty} f(t) \psi_{(s,\tau)}^*(t) dt$$  \hspace{1cm} (2.16)

where \( s \) is scale, \( \tau \) is translation, \( \psi \) is a function, shaping the wavelet (also called the mother wavelet), and * is the complex conjugate symbol. The mother wavelet is continuously scaled and shifted to match the desired output signal. There is no set way to choose the mother wavelet function. It all depends on what the input looks like and what the purpose of the transform is [36].

In image processing, the wavelet transform is used to describe an image with not only its spatial properties but also its frequency properties. In this context, a high frequency would be a part of the image where the pixel intensity vary a lot between its neighbors. Usually, this indicates an edge. The high frequency areas of an image are often the ones that matter the most. If the pixel intensity varies slightly over a large area, chances are that the human eye does not notice any difference. Therefore, it is common in for example image compression to
remove the low frequency components of an image, removing data humans otherwise would not notice.

When implementing the wavelet transform in a practical manner, there is two parts to it, decomposition and reconstruction. The decomposition is usually done by defining a filter bank, which is shown in Figure 2.7. The concept is to high pass-filter and low pass-filter the image into two new downscaled images and then repeat once more for both. To reconstruct the image, a similar pattern was followed, but this time the coefficients of each level were up-sampled to form the original image. Note that this is an approximation of the actual continuous wavelet transform listed in Equation 2.16. This method is called the Discrete Wavelet Transform (DWT) [36].

2.5.8 Rain Removal Using Discrete Wavelet Transform and Bilateral Filtering

A methodology for extracting and removing rain was proposed in 2012 by Xinwei Xue et al. in the report Motion robust rain detection and removal from videos [8]. The core idea was to make a map of all the rain and then remove the rain by extrapolating the parts of the image where rain was detected. The extrapolation was done by image inpainting. The key for getting this to work well was how they generated the rain map. Image inpainting yields worse results and is more computationally intensive the larger the area to extrapolate is. It is therefore of importance to keep the rain map as precise as possible [8].

To extract the rain, several maps were combined. First off, wavelet domain feature extraction was done. This was to detect high frequency components of the image. This was done by doing the DWT decomposition, but before the image was reconstructed, the low frequency parts were discarded. The end re-
result was then binarized. This yields a map of all high frequency components of the image [8].

The second map was similar to the first one, but before the DWT feature extraction was done, the image was filtered using bilateral filtering. The thought behind this was to get a map only containing the high frequency components of an image that are not caused by rain or noise, but only those caused by actual edges of objects in the image. This could be achieved because of the edge preserving properties of the bilateral smoothing. This map was also binarized. By subtracting the bilateral filtered map from the not filtered, mainly the edges caused by noise and rain could be extracted. This map is called a detailed edge map [8].

The final map that was created was a so called motion map. To reduce invalid detection, the idea was to not consider parts of the video that were motionless. Therefore a comparison was done between each frame, to recognize rapid increases in intensity for each frame. Each pixel was evaluated according to Equation 2.17 [8].

\[ I_m - I_{m-1} \geq c \] (2.17)

From the equation, a map can be formed that includes all pixels whose intensity between two frames are equal to or above the threshold c. A chart of all steps of the rain map extraction can be seen in Figure 2.8.
This recognizes a lot of rain because of its spatial properties. In most videos, the frame rate is so low in relation to the speed of the rain that a rain-drop usually does not occupy the same pixel two frames in a row [8].

To generate the final rain map, the intersection of the detailed edge map and the motion map was generated. The results that was achieved in their report can be seen in the report by Xinwei Xue et al. [8].

### 2.6 Object Detection

Object detection is a branch of machine learning with many different methods and is an area that is growing rapidly. There are many ways to detect and recognize objects in images and the solutions have different trade-offs in regards to accuracy and latency. Examples of methodologies would be:

- Using Haar-Like Features to recognize objects [39].
• Naive Bayesian Object Tracking [40].
• Object Recognition with Neural Networks [41].

In the scope of this thesis, it is only necessary to cover the Neural Network based algorithms which will be described below.

2.6.1 Artificial Neural Networks

Artificial Neural Networks (ANN) are, as the name suggests, a model with similarities to the real life neural networks (i.e. the human brain). While a biological neural network has physical neurons, an ANN has virtually created neurons in its software. Each neuron has an activation function. The activation function can be defined by the creator to any function that, depending on the signal it receives, outputs a signal or not, see Figure 2.9 [42].

![Figure 2.9: A single neuron](image)

The neuron in itself does provide many means for learning. It is a combination of many connected neurons, often in layers along with so called weights. The network works by connecting multiple neurons and multiplying their signals with a weight for each connection, see Figure 2.10. Training the network would then be tuning these weights to give a desired output if a specific input is given [42].
2.6.2 Convolutional Neural Networks

As ANN’s started to be used for classifying images, there were some issues that arose. First off, the previous ANN models did not scale well with the resolution of the images and secondly the network would always look at the image in its entirety. Convolutional Neural Networks (CNN’s) was proposed as a solution to this [44].

While a typical ANN take all the pixels of the image and connects them to the network a CNN works by sliding (convolving) a kernel over the image, taking the dot product of the kernel and the underlying pixels for each position. This produces a so called activation map which contains the results of the products of every possible position in the image.

In a CNN, the values of the kernel are the weights. The results in a trained CNN are multiple of these kernels, that can activate a neuron if it detects a specific feature at some position of the input. This is done in multiple layers. If the input image to the network is an image of a face, the first layers of the kernels might activate on edges. The kernels a couple of layers deeper might activate on mouths or any other more complex features. To classify objects, CNN’s usually have a final, fully connected layer, which is illustrated in Figure 2.11 [30].
2.6.3 You Only Look Once

One large aspect of computer vision is time. The state of the world is constantly changing and objects are moving. To make good decisions based on computer vision systems, low latency is key. Today there are many networks that specialize on just that, one of them being You Only Look Once, or YOLO.

YOLO takes a slightly different approach than other networks. YOLO starts out by dividing the input image into a predefined amount of cells, see Figure 2.12. It then proceeds by using a trained network to predict a set amount bounding boxes per cell, see Figure 2.13 [12].
The model then predicts with what certainty each bounding box contain an object and what object it contains. This is done via a relatively standard CNN. All bounding boxes with a classification confidence level above a user defined limit is then the final detected bounding boxes, see Figure 2.14. The power of YOLO is in the prediction of the bounding boxes. Since the bounding boxes are predicted, the image only has to be fed through the network once, thus the name You Only Look Once. This is also what makes it fast compared to many other neural networks. This method is close to how humans detect objects which is simultaneously with different confidences [12].

Figure 2.13: YOLO grid with predicted bounding boxes. Thicker border indicate stronger confidence [12].
2.7 GPU-Accelerated Computing

One of the key pieces of hardware that allows deep learning and image processing to perform in real time is the Graphics Processing Unit (GPU) [46]. GPU’s were designed to accelerate creation and altering of images. In more detail, what had to be sped up was large matrix operations.

The core principles of a GPU is that all its computations are done massively in parallel. This is achieved by the many cores of a GPU which as of today can be up to thousands in a single unit. This level of parallelism is used to rapidly do many calculations simultaneously for each GPU cycle. The instructions of a GPU is based on the Single Instruction Multiple Data (SIMD) paradigm. The reasoning is that a single instruction is given to all threads that then act on multiple parts of data at once. In images it is for example common to apply a filter. With a GPU, this can be done for each pixel at once (assuming the image has lesser amount of pixels than the GPU has threads) [47].

Nvidia’s toolkit for developing general applications with GPU’s is named CUDA which supports C/C++, Python, Matlab and Fortran and includes GPU-accelerated libraries, a compiler, developer tools and CUDA runtime [48].
2.8 Related Work

As of today there are many different sources that have proposed different solutions to remove rain from images and videos. This section will briefly discuss other solutions and also compare their test cases.

The two methods used for this thesis are proposed by Xinwei Xue et al. [8] and Wan-Joo Park et al. [13]. The method first mentioned use wavelet transform and bilateral filtering along with image inpainting for rain removal. The latter use Kalman filtering and spatio temporal properties of rain.

The first report show an example where object tracking is tested with and without rain and better results are yielded without rain. Their conclusion discuss better results when using two frames with moving objects compared to other solutions. The presented testing is only qualitative where frames are compared with and without rain removal. The second algorithm uses Kalman filtering for estimating new pixel intensity values and conclude that it can be implemented in real time processing. Their solution is also only qualitatively evaluated with video filmed with a stationary camera and no moving objects. The report states expectations of how the Extended Kalman Filter would be more robust for various environments.

Abdel-Hakim’s proposal [49] uses low-rank recovery where both static and dynamic frames are tested with real rain and rain added in post processing. The author concludes the results are better than other solutions with qualitative testing. Quantitative testing is also provided where pixel intensity errors are measured with how many frames are used for comparison where the tested frames are simulated with added rain in post processing. Kalia and Jaikar [50] use temporal-spatial statistical properties to remove rain and test their method qualitatively and quantitatively where resolution and time taken are presented.

Chen et al. [51] propose an algorithm that works with a fast moving camera. The algorithm is based on super positioning and convolutional neural networks. Their conclusion is that their solution provide better results than state-of-the-art methods. Quantitative testing is done for execution times with different CNN’s. Reconstruction methods are also tested with quantitative methods.

A paper from Luo et al. [52] discusses a solution that works with only a single
frame, which uses discriminative sparse coding built on a nonlinear generative model. Their conclusions are based on results outperforming other methods with both real and synthesized rain. Testing is done both qualitatively and quantitatively where peak signal to noise ratio and structural similarity are evaluated.

Huang et al. [53] also propose a solution that works with single frames that learns context information with support vector machines and principal component analysis and later on filter out segmented rain. Both qualitative and quantitative tests were done where the second method consisted of peak signal to noise ratio compared to other methods and got better results than other single-image based algorithms.

2.8.1 Conclusion

The methods described earlier in Section 2.8 all provide their own proposals for removing rain but none of them discuss use cases or any bigger contexts than the actual algorithms. This thesis differentiates from the other methods by evaluating if a failure mode for a camera sensor could be improved. Two algorithms were tested in the context of an end to end system for an autonomous vehicle where object detection was tested with different video resolutions and lighting conditions, and accuracy and latency trade-offs was evaluated.
Chapter 3

Implementation

3.1 Vehicle Dynamics Simulation

In order to pose requirements on the latency of the rain removal algorithms, simulations were done of the braking distances of a car. Two stopping distances were arbitrarily chosen to a maximum allowed distance of 80m and 30m. These were used as reference for the upcoming results. To simulate stopping distances for an AV in as realistic circumstances as possible parameters from an actual car was collected. For this thesis a Tesla Model S P100D was used. The simulation was done on Mathworks Matlab Simulink where road/tire dynamics, wheel dynamics and vehicle dynamics were calculated with friction coefficients from rainy conditions to fit the purpose of this thesis. The simulation drawing can be found in Figure 3.1 and all parameters can be found in Table 3.1 with a few of the parameters arbitrarily chosen. The simulation was inspired by Mathworks [54] and work by Anthony Stark [55]. The stopping distance is simulated with Anti-lock Braking System (ABS) and wheel slip is considered for more realistic conditions. The stopping distance can be calculated by adding the actual braking distance and the delay from noticing an obstacle to physical braking. The theory behind the vehicle dynamics simulation and effects of latency were presented in Section 2.2.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description of parameter</th>
<th>Value [Unit]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_w$</td>
<td>Wheel radius [56]</td>
<td>0.241 [m]</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Air density</td>
<td>1.225 [kg/m$^3$]</td>
</tr>
<tr>
<td>$\mu_s$</td>
<td>Rolling resistance coefficent</td>
<td>0.025</td>
</tr>
<tr>
<td>$J_w$</td>
<td>Moment of inertia</td>
<td>0.3828 [kgm$^2$]</td>
</tr>
<tr>
<td>$CdA$</td>
<td>Drag area [57]</td>
<td>0.576 [m$^2$]</td>
</tr>
<tr>
<td>$M_v$</td>
<td>Vehicle mass [57]</td>
<td>2170 [kg]</td>
</tr>
<tr>
<td>$N_w$</td>
<td>Number of wheels during braking</td>
<td>4</td>
</tr>
<tr>
<td>$T_{\text{max}}$</td>
<td>Maximum braking torque</td>
<td>2000 [Nm]</td>
</tr>
<tr>
<td>$T$</td>
<td>Time constant</td>
<td>0.01 [s]</td>
</tr>
</tbody>
</table>

Table 3.1: Parameters of Tesla Model S P100D for simulation of stopping distances.

![Simulink model for ABS braking.](image)

Figure 3.1: Simulink model for ABS braking.

### 3.2 OpenCV

To aid the implementation, the algorithms used were implemented with the help of OpenCV. OpenCV is an open source software library with its functionality being focused on handling image and video. OpenCV is generally
considered a standard when developing software for computer vision applications. OpenCV also provide a library for easy usage of Nvidia CUDA for running large matrix operations in parallel on the GPU. The OpenCV library is available for multiple languages such as Python, C++, Java and Matlab and is written in C++ [58].

3.3 YOLOv3

The object detection algorithm used for this thesis was You Only Look Once (YOLOv3) which was pre-trained through the Darknet [29] framework on the ImageNet [59] dataset. Both YOLO and Tiny YOLO were used for testing where Tiny YOLO is a smaller network which works faster but is less accurate. The theory regarding YOLOv3 was introduced in Subsection 2.6.3.

3.4 Rain Removal Using Kalman Filter

In order to test the Kalman filter algorithm, it was first implemented with Python and OpenCV for making sure the algorithm could perform up to the standards the report described, even if it would do so slowly. When confirmed, it was translated to C++ in order to make use of the graphics card available with Nvidia CUDA and OpenCV combined. With the CUDA optimizations, parallel computations sped up the matrix calculations greatly.

The parameters that were tuned for optimal behaviour were the measurement noise covariance which was set to $\mathcal{N}(0, 5)$ and the process noise covariance which was set to $\mathcal{N}(0, 50)$. These values were set based on the report by Park et al. [13].

Matrices such as $A_{t-1}$ and $H_t$ were chosen as identity matrices, based on the report from Park et al. [13], since each pixel from the same frame are independent from each other. The covariance matrix $\Sigma_{t-1}$ was also set to an identity matrix, making it possible to set the Kalman gain as a scalar instead of a large quadratic array with the same values on the diagonal. Since the Kalman gain is calculated with an inverse matrix, which has the time complexity $\mathcal{O}(n^3)$, the choice for a scalar was much more time efficient.

The expected measurement was chosen as the wanted pixel intensities from the frame one time step earlier while the predicted measurement was the cur-
rent frame with high intensity pixels which were chosen for descent.

Since there are many pixels not affected by the increased intensities it was important to only filter out the ones that needs to. Differences in pixel intensities between frames were first calculated with CUDA to later be available to create a threshold mask array, where the thresholds can be tuned for adjusting the sensitivity of the filter. When the frame is masked with the threshold array it can be used in the Kalman filter algorithm. The output array can then be masked with the threshold array to filter out the pixels that should not be altered with added noise. This array can then be added with an array only containing the pixels that should not be altered which produces the final filtered frame.

### 3.5 Rain Removal Using Discrete Wavelet Transform and Bilateral Filtering

The code for the wavelet transform and bilateral filtering method was implemented in C++. When reading images and accessing the intensity values of the pixels, OpenCV was initially used to do every step of the algorithm. In doing the DWT, pixels were initially accessed individually via OpenCV. The wavelet chosen for the DWT was the Haar wavelet because of its simplicity. Implementing this yielded a working but slow implementation running entirely on the CPU.

As the first iteration was finished the next step was to get it working in a more real-time manner. Many of the necessary operations, such as bilateral filtering, were already implemented for CUDA GPU’s in OpenCV. However, there were also several functions that were not implemented in OpenCV that were necessary. For example there was at the time of this report no implementation for the Discrete Wavelet Transform and its inversion in OpenCV. The code for this was therefore implemented and compiled in CUDA, and then linked to the rest of the project. OpenCV provides options for accessing the raw image data on the heap for these kinds of implementations which made custom CUDA functions easier implement.
3.6 Testing

While doing measurements it turned out to be hard to compare results without comparing them to a common ground truth. Without a ground truth, one possibility would be to use different videos with rain and without, measuring how the average amount of classifications vary. The problem is that it would not necessarily represent the actual performance of the object detection since false positives would increase the performance metric. Furthermore, no information regarding the precision of the bounding boxes would be measured. By using a ground truth, it was possible to measure the accuracy of the bounding boxes while also minimizing the effects of false positives.

By comparing a The problem with this was that it was hard to find video footage satisfying the requirements. The plan was to get a ground truth by running YOLOv3 on a video without any rain. The testing would then be carried out by running rain removal and YOLOv3 on the same video but with rain. The chances of capturing an exact sequence of moments with same conditions both with and without rain are close to impossible. Finding two almost identical videos, one with rain and one without, were also not possible. It was therefore decided that a good option was to add rain in post production. This was done using a pre-keyed video of actual rain and overlaying it over the ground truth video. While computer simulations could have been used to generate the rainy footage and the ground truth, adding rain in post production was preferred as it looked more realistic. The result can be seen in Figure 3.2 and the reason why adding rain in post production is reliable is discussed in the Discussion chapter. Testing was done on two videos which included vehicles and people. One video was filmed during daytime [60] and the other one during the evening [61].
3.6.1 Combining Rain Removal with the Object Detection

The filtered videos were rendered in different resolutions for them to be run through the YOLOv3 object detection algorithm. A test tool was created for this thesis in Python that exports text files where information such as confidence, label, frame iterations and an objects position, could be sorted structurally for convenient visualization. Graphics of the results were also created in this testing tool, which are referred to in the Results chapter.

Testing is done on a license free video from [60] and a frame with object detection is shown in Figure 3.3.
3.6.2 Resolution and Downscaling

The research questions cover testing the algorithms for various resolutions. For repetitive testing it was important to have the same ground truth at all times and to test the algorithms on the same videos but with different video resolutions. It was not in the scope for this thesis to recreate the same live traffic situations, allowing true low resolution recordings, therefore, downscaling the videos was considered more reasonable. Linear interpolation is used for downscaling. Measuring the latency will be described in Subsection 3.6.5. Video resolutions tested can be seen in Table 3.2.

<table>
<thead>
<tr>
<th>Video Resolutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1920 x 1080</td>
</tr>
<tr>
<td>1280 x 720</td>
</tr>
<tr>
<td>960 x 540</td>
</tr>
<tr>
<td>848 x 480</td>
</tr>
<tr>
<td>640 x 360</td>
</tr>
</tbody>
</table>

Table 3.2: Table over tested video resolutions.
3.6.3 Testing the Rain Removal

Regardless of the results from the object classifier, it was considered desirable to test the rain removal by itself to make sure it works somewhat as intended. The best case scenario would be that the derained video looks just like the ground truth. It was therefore decided that a metric for testing the performance of the rain removal, would be the average of comparing by how much each pixel intensity differs from the ground truth, see Equation 3.1. The usage of this metric is to numerically calculate a difference between images instead of a comparison with the human eye.

\[
I_{ad} = \frac{1}{CP} \sum_{c=1}^{C} \sum_{p=0}^{P} \frac{|I_{c,p} - G_{c,p}|}{P}
\]  

(3.1)

\(I_{ad}\) is the average intensity difference, \(C\) is the number of color channels, \(P\) is the number of pixels, \(I\) is the intensity value of the video that is to be compared to the ground truth and \(G\) is the pixel intensity value of the ground truth.

When two videos are compared in this manner, a result of 0 would mean that the videos are identical. The higher the number, the more different the two videos are. To test the rain removal, the footage with its rain filtered out and the footage with solely the rain added was compared to the ground truth.

3.6.4 Measuring YOLOv3 Accuracy

To do classifications with YOLOv3, it was decided that the Darknet framework would be used. Since the Darknet framework does not have any tools for measuring the accuracy, it was decided that the Darknet framework would be modified slightly to store its output data in some persistent file format for later analysis. For simplicity, the data was exported to one CSV file for each video, containing the data that can be seen in Table 3.6.4.

<table>
<thead>
<tr>
<th>Frame No.</th>
<th>Object</th>
<th>Confidence</th>
<th>x1</th>
<th>y1</th>
<th>x2</th>
<th>y2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Car</td>
<td>87%</td>
<td>1430</td>
<td>503</td>
<td>1890</td>
<td>380</td>
</tr>
<tr>
<td>2</td>
<td>Person</td>
<td>20%</td>
<td>300</td>
<td>480</td>
<td>500</td>
<td>620</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\(x1, x2, y1, y2\) are the coordinates for the bottom left corner and top right corner of each bounding box. The data was then parsed, processed and analyzed in a Python script using the matplotlib module for visualization.
The task of measuring the accuracy of the object classifier was not a completely straightforward task. The approach was to measure the total Intersection over Union (IoU) of each frame and then calculate the average IoU over all frames of the video, see Equation 3.2.

\[ IoU = \frac{A_{\text{intersection}}}{A_{\text{combined}}} \]  

(3.2)

By calculating the IoU by comparing ground truth bounding boxes with for example bounding boxes after adding the rain removal algorithms, it can act as a metric of accuracy. The higher the IoU, the closer the tested data is to the ground truth data. A problem encountered however was that for parts of an image with many objects in close proximity, the bounding boxes for different objects could sometimes overlap, leading to a very low IoU, which in turn lowers the average IoU. To deal with this, only objects with an IoU greater than 50% were compared.

Differences in confidence levels were also measured for testing the accuracy. The YOLOv3 algorithm detections are based on a predicted confidence level. If the confidence level is above a user defined threshold it is considered a detection. This confidence level was measured for all videos, both with and without rain, to evaluate the differences. Visualizations of the amount of classifications per frame were also done for possibly giving a better understanding. High confidence level doesn’t however ensure that it is 100% correct, which is leading to some uncertainty. It is however a metric that can be used for insight of what the model believes.

### 3.6.5 Measuring Speed

For the Kalman filter method, the latency was straightforward to measure using Python’s built in time module. The time before and after each image was measured and the difference could then be calculated as the latency, see Equation 3.3.

\[ t_{\text{lat}} = t_n - t_{n-1} \]  

(3.3)

In the case of the wavelet transform and bilateral filtering method, the image inpainting was not implemented on the GPU as it was considered to be too large of a commitment in the scope of this project. This means that the latency of accelerated inpainting on the GPU could not be evaluated. If the inpainting method [62] proposed by Xinwei Xue et al. [8] was implemented on the GPU,
the performance increase would however probably not be substantial as parts of the algorithm has to happen sequentially. While different subsets, $\Omega$, of the mask could be inpainted in parallel, the inpainting of each $\Omega$ would have to be done in a sequential manner since it is required for the algorithm. This means that not only would it be slow in relation to the parts that are completely parallelizable but also that the latency would increase with the size of the $\Omega$’s.

In the result chapter the average latency of the image inpainting will be presented separately from the DWT&BF latency. The information is relevant to keep separate as the latency of solely the DWT&BF could be interesting to someone constructing a rain removal algorithm using the DWT&BF as a way of detecting and mapping the rain but using another method for interpolating the raindrops away.
Chapter 4

Results

4.1 Rain Removal Performance

Visually, the success of the rain removal can be clearly observed, see Figure 4.1.

![Figure 4.1: A: No rain removal, B: Kalman Filter, C: DWT&BF](image)

The average pixel intensity difference for the two rain removal methods and the footage with added rain compared to ground truth can be seen in Table 4.1. The values are based on the method described in 3.6.3.

<table>
<thead>
<tr>
<th>Video</th>
<th>$I_{ad}$ (Day)</th>
<th>$I_{ad}$ (Night)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added rain</td>
<td>6.25</td>
<td>9.27</td>
</tr>
<tr>
<td>Derained with DWT&amp;BF</td>
<td>6.08</td>
<td>5.98</td>
</tr>
<tr>
<td>Derained with KF</td>
<td>5.40</td>
<td>6.53</td>
</tr>
</tbody>
</table>

Table 4.1: Table of the average pixel intensity difference compared to the ground truth.
4.2 Accuracy

Measurements in the metrics of intersection over union (IoU) and confidence levels for the object detection algorithm YOLOv3 were done and the results can be shown in this section. Only the first 50 frames are shown in this chapter. Graphs spanning over all frames of the benchmark video clip are shown in Appendix A. Averages in IoU, confidence levels and amounts of 100% confidence level detections for each method are also shown. At the end of this chapter, tables of how different video resolutions affect the latency, IoU and confidence levels are presented.

4.2.1 Intersection over Union

The ground truth of the IoU was the bounding boxes for the video of traffic without any rain. The IoU for both methods was compared with the same video with added rain in post production. The higher the percentage, the closer the bounding boxes were to the ground truth. In Figure 4.3a the average IoU over different resolutions during day light is shown and in Figure 4.2 the methods compared in 1080p is shown. Figure 4.3b show the average IoU in darker lighting conditions. IoU in different pixel resolutions are presented in Figures A.1a, A.1b, A.2a and A.2b in Appendix A.
Figure 4.2: IoU in percentage for DWT & BF, KF and added rain with ground truth in 1080p. Video is a traffic video in daylight.

It can be seen in Figure 4.3a that the KF reached its maximum average IoU at 480p while DWT & BF and added rain gets higher IoU with increased pixel resolution. Figure 4.3b shows that the average IoU is highest for all resolutions without removing any rain. It is also shown that the average IoU is at its lowest in 1080p for all methods.
Figure 4.3: Average IoU in percentage for KF, added rain and DWT & BF with ground truth in different pixel resolutions.
4.2.2 Confidence Level

The difference in confidence, measured in percentage points, is calculated and shown in Figure 4.4 where the difference is when compared to the ground truth which is without added rain. The calculations for the Figure 4.4 were on videos of traffic in daylight with 1920 x 1080 pixel resolution. The graph shows fluctuations where spikes over zero is common but has averages below zero which can be seen in Table 4.2 and 4.3 and also in Figure 4.5a and 4.5b. Figures over confidence levels for all pixel resolutions are presented in Appendix A. The average confidence level difference for each method in different resolutions are shown in Figure 4.5 and the average amount of 100% confidence detections for each method are shown in Figure 4.6. The average confidence levels of all methods during daylight has a peak at 480p resolution where KF and added rain are above zero and DWT & BF is below zero. From 540p and upwards, all methods average confidence levels are below zero. With less lighting has added rain a peak at 540p while KF and DWT & BF has their highest value at 1080p. The average amounts of detection in 100% confidence levels in daytime are highest in 480p and 1080p where added rain has the highest value except for 720p where DWT & BF has the highest value. In darker scenes has ground truth the most amount of detections for all resolutions while the others methods peak at 540p. The results will be discussed in Chapter 6.
Figure 4.4: Confidence level difference in percentage points for DWT & BF and KF with ground truth and rain with ground truth in 1080p.
Figure 4.5: Average confidence level difference in percentage points for KF, DWT & BF and added rain with ground truth in different pixel resolutions.

(a) Daytime lighting condition.

(b) Evening lighting condition.
Figure 4.6: Average amounts of detection in 100% confidence level for KF, DWT & BF, ground truth and added rain in different pixel resolutions.
4.3 Impact of Resolution

In Tables 4.2-4.8 results for different filtered videos are presented. For each video and filter, the chosen video resolutions are presented with latency per frame, confidence levels and average IoU. The latency for each frame is presented in seconds for both the rain removal algorithms and the object detection, average confidence levels is presented in difference in percentage points and average IoU in percent. For both algorithms and the YOLOv3 object detection, the latency is lowered as the resolution is decreased. In brighter images, KF has increased average confidence in 480p and 540p and highest IoU but doesn’t show a specific trend with increased average confidence and IoU since 360p has higher IoU but less confidence than 720p, even if the numbers are somewhat close. DWT & BF has the lowest confidence for all resolutions but does have a higher IoU than the KF for all resolutions above 480p, which is shown in Figure 4.3a. The impact of resolution in darker images favours toward higher pixel resolutions when comparing confidence levels. The largest pixel resolution shows the lowest IoU value while all other resolutions show similar values.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>KF latency [s]</th>
<th>OD latency [s]</th>
<th>Conf [pp]</th>
<th>IoU [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1920 x 1080</td>
<td>0.155</td>
<td>0.874</td>
<td>-0.432</td>
<td>92.496</td>
</tr>
<tr>
<td>1280 x 720</td>
<td>0.061</td>
<td>0.277</td>
<td>-0.186</td>
<td>92.566</td>
</tr>
<tr>
<td>960 x 540</td>
<td>0.041</td>
<td>0.268</td>
<td>0.017</td>
<td>92.496</td>
</tr>
<tr>
<td>848 x 480</td>
<td>0.038</td>
<td>0.264</td>
<td>0.156</td>
<td>92.569</td>
</tr>
<tr>
<td>640 x 360</td>
<td>0.023</td>
<td>0.258</td>
<td>-0.246</td>
<td>91.880</td>
</tr>
</tbody>
</table>

Table 4.2: Table over results for daytime traffic video with KF and YOLOv3.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>DWT &amp; BF latency [s]</th>
<th>OD latency [s]</th>
<th>Conf [pp]</th>
<th>IoU [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1920 x 1080</td>
<td>0.0545</td>
<td>0.874</td>
<td>-0.663</td>
<td>94.758</td>
</tr>
<tr>
<td>1280 x 720</td>
<td>0.0296</td>
<td>0.277</td>
<td>-0.342</td>
<td>94.086</td>
</tr>
<tr>
<td>960 x 540</td>
<td>0.0226</td>
<td>0.268</td>
<td>-0.458</td>
<td>93.619</td>
</tr>
<tr>
<td>848 x 480</td>
<td>0.0199</td>
<td>0.264</td>
<td>-0.307</td>
<td>93.101</td>
</tr>
<tr>
<td>640 x 360</td>
<td>0.0157</td>
<td>0.258</td>
<td>-0.707</td>
<td>91.046</td>
</tr>
</tbody>
</table>

Table 4.3: Table over results for daytime traffic video with DWT & BF and YOLOv3.
### Table 4.4: Table over results for traffic video during evening lighting condition with KF and YOLOv3.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>KF latency [s]</th>
<th>OD latency [s]</th>
<th>Conf [pp]</th>
<th>IoU [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1920 x 1080</td>
<td>0.155</td>
<td>0.874</td>
<td>-1.615</td>
<td>82.610</td>
</tr>
<tr>
<td>1280 x 720</td>
<td>0.061</td>
<td>0.277</td>
<td>-1.781</td>
<td>87.028</td>
</tr>
<tr>
<td>960 x 540</td>
<td>0.041</td>
<td>0.268</td>
<td>-1.850</td>
<td>86.160</td>
</tr>
<tr>
<td>848 x 480</td>
<td>0.038</td>
<td>0.264</td>
<td>-1.800</td>
<td>87.338</td>
</tr>
<tr>
<td>640 x 360</td>
<td>0.023</td>
<td>0.258</td>
<td>-2.330</td>
<td>86.558</td>
</tr>
</tbody>
</table>

### Table 4.5: Table over results for traffic video during evening lighting condition with DWT & BF and YOLOv3.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>DWT &amp; BF latency [s]</th>
<th>OD latency [s]</th>
<th>Conf [pp]</th>
<th>IoU [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1920 x 1080</td>
<td>0.0545</td>
<td>0.874</td>
<td>-1.795</td>
<td>86.161</td>
</tr>
<tr>
<td>1280 x 720</td>
<td>0.0296</td>
<td>0.277</td>
<td>-2.372</td>
<td>91.028</td>
</tr>
<tr>
<td>960 x 540</td>
<td>0.0226</td>
<td>0.268</td>
<td>-2.731</td>
<td>90.708</td>
</tr>
<tr>
<td>848 x 480</td>
<td>0.0199</td>
<td>0.264</td>
<td>-2.111</td>
<td>90.202</td>
</tr>
<tr>
<td>640 x 360</td>
<td>0.0157</td>
<td>0.258</td>
<td>-2.890</td>
<td>89.328</td>
</tr>
</tbody>
</table>

### Table 4.6: Table over results for daytime traffic video with KF and Tiny YOLOv3.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>KF latency [s]</th>
<th>OD latency [s]</th>
<th>Conf [pp]</th>
<th>IoU [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1920 x 1080</td>
<td>0.155</td>
<td>0.097</td>
<td>-1.390</td>
<td>9.881</td>
</tr>
<tr>
<td>1280 x 720</td>
<td>0.061</td>
<td>0.049</td>
<td>-1.570</td>
<td>11.752</td>
</tr>
<tr>
<td>960 x 540</td>
<td>0.041</td>
<td>0.038</td>
<td>-1.549</td>
<td>11.470</td>
</tr>
<tr>
<td>848 x 480</td>
<td>0.038</td>
<td>0.035</td>
<td>-1.643</td>
<td>10.151</td>
</tr>
<tr>
<td>640 x 360</td>
<td>0.023</td>
<td>0.030</td>
<td>-1.738</td>
<td>5.817</td>
</tr>
</tbody>
</table>

### Table 4.7: Table over results for daytime traffic video with DWT & BF and Tiny YOLOv3.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>DWT &amp; BF latency [s]</th>
<th>OD latency [s]</th>
<th>Conf [pp]</th>
<th>IoU [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1920 x 1080</td>
<td>0.0545</td>
<td>0.097</td>
<td>-1.355</td>
<td>9.395</td>
</tr>
<tr>
<td>1280 x 720</td>
<td>0.0296</td>
<td>0.049</td>
<td>-1.649</td>
<td>11.875</td>
</tr>
<tr>
<td>960 x 540</td>
<td>0.0226</td>
<td>0.038</td>
<td>-1.746</td>
<td>14.112</td>
</tr>
<tr>
<td>848 x 480</td>
<td>0.0199</td>
<td>0.035</td>
<td>-1.610</td>
<td>9.043</td>
</tr>
<tr>
<td>640 x 360</td>
<td>0.0157</td>
<td>0.030</td>
<td>-1.764</td>
<td>7.795</td>
</tr>
</tbody>
</table>
Table 4.8: Latency of image inpainting.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Inpainting avg latency [s]</th>
<th>Max latency [s]</th>
<th>Min latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1920 x 1080</td>
<td>8.48</td>
<td>16.31</td>
<td>3.75</td>
</tr>
<tr>
<td>1280 x 720</td>
<td>2.59</td>
<td>4.46</td>
<td>1.06</td>
</tr>
<tr>
<td>960 x 540</td>
<td>0.78</td>
<td>2.03</td>
<td>0.15</td>
</tr>
<tr>
<td>848 x 480</td>
<td>0.68</td>
<td>1.51</td>
<td>0.14</td>
</tr>
<tr>
<td>640 x 360</td>
<td>0.27</td>
<td>0.71</td>
<td>0.04</td>
</tr>
</tbody>
</table>
Chapter 5

Evaluation

This chapter treats discoveries done during the course of the project which does not qualify as results but still are relevant in this thesis.

5.1 Reliability of Ground Truth

For evaluation of both the Intersection over Union and the confidence levels, a ground truth was used. This ground truth is the objects that YOLOv3 detects from the original video without rain. A possible issue with this was that the YOLOv3 algorithm is not perfect, causing imperfections in the ground truth. This makes it difficult to know if objects in the ground truth are not detected or if there are false positives. Since the comparisons in the report are all made through the prediction algorithm, the ground truth could however be considered valid, at least for measuring performance. If the rain removal algorithms or added rain would cause even better predictions, this would likely coincidences or by chance. Adding rain to the video should not increase the performance of the object detection. Therefore the ground truth can be used to draw conclusions in a relative sense.

5.2 Night Time Rain Removal

One interesting discovery was how the rain removal seems to operate more effectively during night time. The results of this can be seen in Figure 5.1, 5.2 and 5.3.
Figure 5.1: Night time frame without any filtering [63].

Figure 5.2: DWT & BF processed frame at night time. The result with image inpainting can be seen on the left and the rain map generated can be seen on the right [63].

Figure 5.3: KF processed frame at night time [63].
As seen in the figures, the rain appeared to be removed very well. This seemed to imply that the contrast between the rain and the dark background created by the headlights reflecting off the raindrops, causing the rain to be more efficiently removed. During this thesis, no night time video with objects in it was discovered. This resulted in night time object detection not being possible to test. Further investigation on this is recommended as future work.

5.3 Evaluation of Snow Removal

In investigating the limits of the proposed algorithms, several edge cases were tested. For example the rain removal was tested in both daylight and nighttime lighting conditions. Another interesting condition to test was whether the filtering would work well during snowfall. Snowfall differs a lot from rain in regards of the physical properties. The snowflakes are usually larger than raindrops, while falling at a slower speed. This proved to have a big impact on the effectiveness of the rain removal.

For the DWT & BF method, the snow removal was largely unsuccessful. While the motion map maps the snowflakes relatively well, DWT edge mapping does not correctly map the snowflakes even when experimenting with tuning of the parameters. This is largely because the DWT edge mapping detects edges. Because of the larger size of most of the snowflakes, only the edges are detected and the center part is left out of the map, see Figure 5.4. This leads to poor performance.

![Figure 5.4: Demonstration of DWT edge mapping only detecting the edges of snowflakes. The original is to the left and the map is on the right.](image)

With the KF method, only the smaller snowflakes are successfully filtered out while the larger ones can not be removed entirely. The reason why is that each
large snowflake covers the same pixel for multiple frames and since KF uses the previous frame as a reference, the background colour would not be found thus not allowing for good interpolation.

### 5.4 Object Detection Latency

Depending on what is required from the object detection system there are two versions of YOLOv3. Tiny YOLOv3 is a less complex network, making it faster compared to the full YOLOv3 which should have higher accuracy in its detection. With the graphics card available for this thesis, the achieved frame-rates for YOLOv3 is 1.15 fps in 1080p and 3.73 fps in 540p. Tiny YOLO produces a frame-rate of 10.3fps in 1080p and 26.3fps in 540p. A higher frame-rate was wanted for systems with low latency requirements and can be achieved with more powerful graphics cards and CPU’s. With the results shown in Chapter 4, Tiny YOLO was not considered accurate enough to be evaluated further in this thesis.

### 5.5 Hardware Requirements

#### 5.5.1 Vehicle Scenarios

With the latency’s yielded from the GPU at hand, they can be used to simulate braking distances for a specific car model in rainy weather with wet asphalt. Driving a Tesla Model S P100D in 110km/h, or 30.6m/s, with a camera pixel resolution of 1920x1080 results in driven 31.5m with the 1.029s latency, before activating the braking system, if Kalman filter and YOLOv3 was used.

In order to decide what hardware requirements are necessary two example scenarios needs to be set up. In the first scenario, an object suddenly appears 80m in front of the vehicle. This object could be an animal, debris etc. It is likely that the vehicle will have a velocity of 110km/h or 90km/h. In the second scenario the vehicle will be driving in a crowded city. The vehicle will be driving in 50km/h or 30km/h and an object will appear 30m ahead. The object could either be a vehicle, a human, red light at an intersection or debris. This subsection will evaluate what hardware is needed to be able to make a decision and brake in time. Stopping distances and time to full stop with the chosen parameters from Section 3.1 are shown in Table 5.1 and Figure 5.5. Table 5.1 also shows the initial velocity $v_0$ and what the maximum latency should be in
order to not exceed the stopping distances from the earlier scenarios.

Figure 5.5: Stopping distances with $v_0$ as starting velocity. X-axis shows time to full stop and Y-axis shows braking distance in meters.
\[ r_0 \text{ [km/h]} \quad \text{Time [s]} \quad \text{Stopping distance [m]} \quad \text{Maximum latency [s]} \]

| 110 | 5.3 | 73 | 0.229 |
| 90  | 4.6 | 55 | 1     |
| 50  | 3.3 | 24 | 0.432 |
| 30  | 2.5 | 11.7 | 2.196 |

Table 5.1: Table over results for daytime traffic video with KF and YOLOv3.

### 5.5.2 Computational Requirements

In construction of a fully functional rain removal and object detection system, several conclusions could be drawn during the project regarding the necessary components and performance specifications. This section aims to explain and state these in order to aid future construction of a full rain removal/object detection system in the context of an autonomous vehicle.

Overall, there were a lot of operations with large time complexities if performed in a fully sequential manner. This is common in image processing scenarios. For example, brute force bilateral filtering has a time complexity of \( O(n^2) \), where \( n \) is the amount of pixels in the image. Furthermore, there were many operations, such as the motion map generation, with a minimum time complexity of \( O(n) \) as each pixel of the image had to be read and possibly modified. Because of the large amount of pixel operations, due to the nature of image processing, it was considered a requirement to use a GPU. The GPU used in this thesis was an Nvidia Quadro P1000 with 640 CUDA cores and 4GB of on-board memory. It can operate at a core speed of 1493-1519 MHz and memory speed of 6008 MHz.

In order to evaluate what hardware requirements are needed, Table 4.2 and 4.3 was used. From the requirements in Subsection 5.5.1, which states that the car must start breaking within 0.23s, necessary hardware requirements could then be estimated.

As clearly seen, YOLOv3 running on the GPU at hand is not matching the given requirement as its average latency varies from 0.258s up to 0.874s when scaling the resolution between 360p and 1080p. For both the Kalman filter and the DWT & BF method, the latency is low, making YOLOv3 the limiting factor.
According to the report written by Redmon, Joseph [12] YOLOv3 does run with an average latency of 0.051s at resolution 608x608. This means that a conclusion could be drawn that with a Titan X, the latency would be well below the acceptable limit if the resolution was around 608x608. This is because the latency of the Kalman filter at this resolution would land between 0.023-0.038s and the latency of the DWT & BF would be somewhere between 0.0157s and 0.0199s. Adding the upper bounds with the YOLOv3 latency is well below 0.23s. While higher resolutions probably would run fast enough as well, there was no way of testing it during the course of this thesis as there was no access to a GPU as powerful.

Finally there were also some memory requirements. When the rain removal was running, images had to be loaded into random access memory, both on the GPU and the CPU. This means that depending on the resolution, the amount of memory required for a single image, in bytes, is given by Equation 5.1.

\[
S_{\text{min}} = 3wh
\]

(5.1)

Where \( w \) is the width of the image, \( h \) is the height of the image. The 3 comes from the image being three colors. Since the images are of 8-bit depth (each intensity value is between 0-255) each pixel also occupies one byte. This means that for 1080p video footage there has to be at least 6.22 MB of RAM memory and GPU VRAM memory for a full image to be loaded into the GPU memory. For both methods, at least two images had to be in memory at once for making the motion maps. This meant that the total memory required was at least twice as large (12.45 MB). Another aspect of the KF method for removing rain was that several functions were based on randomizing values for Gaussian noise. This noise was generated by a function running on the CPU. In timing the shuffle process for different matrix sizes with the CPU, an approximately linear time complexity was measured. This could be reduced if it was implemented on the GPU. This process might become a bottleneck for the algorithm and a GPU implementation could improve the latency.
Chapter 6
Discussion and Conclusions

This chapter covers a general discussion and conclusion for this thesis and suggestions for future work are also presented.

6.1 Discussion

For validation purposes it is important to use test cases that are reproducible. However, in this scenario it is impossible to gather videos of exactly the same content both with and without rain. In order to solve this problem rain has been added in post production in order to test the object detection algorithm on a rain free ground truth and the same footage with rain added. Since the accuracy measurements are based on the position and size of the bounding boxes, testing one video without rain and comparing it with another video with rain would not yield comparable results. Adding rain in post production makes comparison of the videos possible since the only difference between the videos is the rain. It was considered to evaluate videos with heavy rain already in them and placing bounding boxes manually. This would however not be perfectly accurate since the bounding boxes wouldn’t be the same size as YOLOv3 would put them. It would also be necessary to place multiple boxes on multiple frames.

One interesting point to discuss are the metrics used for measuring the accuracy. The difference mainly lies in what is desired to evaluate. In most cases, you want as precise bounding boxes as possible, just enclosing the object detected. In this scenario the intersection over union is a relevant metric to evaluate for each bounding box. For the evaluations in the report, the YOLOv3 running on rain free footage is what is considered as ground truth.
Another metric that was used was the object detection confidence level. It is the confidence that YOLOv3 predicts for each bounding box, stating to what certainty the bounding box is perceived to contain a certain object. This metric could say a lot about how the model thinks it is performing but does not compare to any ground truth. An example of where this can cause non-intuitive results can be seen in Figure 4.5a, where the confidence level at 720p is higher for the Kalman filter method but when looking at the IoU for the same resolution, in Figure 4.3a the IoU is instead higher for the DWT & BF method. In darker lighting conditions does similar results show, where in same resolutions is KF more accurate in confidence level but less accurate in IoU.

The performance of accuracy for both methods differ depending on metric. In daylight is the average confidence level higher with KF for all resolutions compared to DWT & BF, however does the later perform better than KF on IoU from 480p resolution and upwards. While the average IoU for DWT & BF increases as the resolution is increased, the KF method performing at its best in 480p and then decreases with increased resolution. In the darker video is DWT & BF better than KF, but worse than rainy footage, in IoU for all resolutions. Results show that YOLOv3 provides better results in brighter footage compared to darker ones. The IoU graphs show more reliable results compared to the confidence graphs that fluctuate between resolutions. The resolution might affect the accuracy for objects further away, however is the risk of missing large object close to the camera much lower. Research on how rain affect the detection of only object further away should be done.

In Section 2.2.2 Effects of Latency it is brought up that computers in AV’s react in around 0.011-0.2s from detection to initialized braking. Considering this as a performance goal, the methods tested in this thesis should not exceed the upper boundary of 0.2s. With the graphics card available for this thesis none of the tested methods are acceptable since YOLO takes 0.258s for computing each frame in the lowest tested resolution of 640 x 360. However, with an Nvidia Titan X, classification with the YOLOv3 network supposedly only takes 0.022s for each frame in 320 x 320 and 0.051s in 608 x 608 according to Joseph Redmon [12]. While no guarantees can be made that resolutions higher than this does work within the given boundary it is very likely that with a powerful GPU such as the Titan X, the latency will be within the acceptable range. Tiny YOLO is always faster than 0.2s with both rain removal methods except for KF in 1080p but is presumably also below the limit with a more powerful
GPU such as the Titan X. However, the results with Tiny YOLO show that the accuracy is far worse than full sized YOLO, making it close to useless for autonomous vehicles or any other accuracy critical scenario.

With the traffic scenarios presented in Section 5.5.1, the maximum latency is 0.23s which YOLO cannot provide with the available GPU and the resolutions tested. However with an Nvidia Titan X it is assumed to be powerful enough to run YOLOv3 and rain removal algorithms with pixel resolutions of 640 x 360 and 848 x 480, given the latency results from Redmon, Joseph [12].

With the results that are received, see Chapter 4, it can be seen that for the probably most relevant metric, the IoU, both the Kalman filter and DWT & BF methods are actually performing worse than if the object detection is simply run on the rainy footage. This is pretty much consistent for all resolutions except for at 360p where the Kalman filter is performing better than the rest. It is however also worth noting that the differences are only differing slightly, about ±2 percentage points. While there is a possibility that the IoU is lower for the footage with the rain removed because it is performing better than the ground truth, it would most likely be based on luck or false positives. This is because predictions including extrapolated pixels simply should not perform better than the original footage without the rain. One reason why YOLOv3 performed well in rainy footage could be because the training data might have included images containing rain. Since the training dataset is too large, it is difficult to know what kind of images were included.

It is interesting to discuss the edge cases of the two rain removal methods. In what scenarios do they fail and when do they perform at best. Both daytime and evening time scenarios were evaluated in the report and as could be seen in the results, the performance in daytime lighting was better. This is however not very surprising as the diffuse lighting of the video yields more detail. Regarding the snow scenario, the size of the flakes were a problem for both methods. Further edge cases could be researched like rain hitting the lens, objects obstructing the lens and varying amounts of rain greatly. It could be interesting to compare the performance in very light rain to heavy rain.

While the effects of the rain removal can be visually perceived and also measured, see Section 4.1, to look more like the original rain free footage, it is pretty clear that in the case of object detection using YOLOv3, the use is likely not of any help. The algorithm performs just as well without the rain removal.
It would however be interesting to further investigate the night time scenario as the results look even more promising with the contrast of the lit raindrops at night time.

6.2 Conclusion

In comparing the latencies of the two rain removal methods, DWT & BF is much faster than KF. The difference in speed between these two methods does increase with pixel resolutions. These latencies are an addition to the latency caused by the YOLOv3 object detection. The latency increase faster for KF with increased pixel resolution than for DWT & BF. The accuracies are similar between resolutions, with average values between 91-95% in IoU and -0.7pp and 0.16pp in difference in confidence levels. KF has the highest IoU and best confidence level in 480p in both daylight and darker footage while DWT & BF has highest IoU in 1080p in daylight and 720p in darker footage. Confidence level is highest in 480p in daylight and 1080p in darker footage. Results show that the rain removal methods are not providing valuable improvements for YOLOv3.

The optimal resolution for a camera with maintained accuracy is 848 x 480 for Kalman filter with 0.156pp in confidence difference and 92.569% in IoU in daylight and -1.800pp and 87.338% in darker footage. For DWT & BF is 960 x 540 optimal with -0.458pp and 93.619% in daylight and 848 x 480 darker footage with -2.111pp and 90.202%.

The final conclusion of this thesis is that these algorithms work well when comparing frames but not well enough on numerical levels in order to improve autonomous driving during rainy weather during the day. Night time scenarios with object detection have however not been tested and is recommended for future work since impressive results have been achieved when comparing frames.

6.3 Future Work

In the following list suggestions for future work will be provided:

- It would be interesting investigating if the blur caused by rain at a far distance can be processed to further increase the view range. This failure mode was neglected in this thesis.
• Use other tools of measurements and evaluate how other methodologies would affect the results.

• Test the rain removing algorithms with different types of object detection algorithms for a more diversified methodology and evaluate if noise could be a factor why YOLOv3 has better confidence levels in rainy weather for some pixel resolutions.

• Visual odometry with SLAM should be tested with rain removing algorithms in order to evaluate if landmark recognition and object detection would be affected.

• Other aspects of detection could be tested with rain removal algorithms such as lane detection during day and/or night.
Bibliography


Appendix A

Results
Figure A.1: IoU in percentage for KF or DWT & BF with ground truth in different pixel resolutions. Video is a traffic video in daylight.
Figure A.2: IoU in percentage for KF or DWT & BF with ground truth in different pixel resolutions. Video is a traffic video in evening lighting condition.
APPENDIX A. RESULTS

Figure A.3: Confidence level difference in percentage points for KF or DWT & BF in different pixel resolutions. Video is a traffic video in daylight.
Figure A.4: Confidence level difference in percentage points for KF or DWT & BF in different pixel resolutions. Video is a traffic video in evening lighting condition.