Detecting and Preventing Drowning Accidents using Thermal Cameras

Søren Bonderup and Jonas Olsson

Master’s Thesis
VGIS10 - 16gr1010 - 2/6 2016
Title: Detecting and Preventing Drowning Accidents using Thermal Cameras

Abstract: Since 2008, 8 persons have drowned in Limfjorden, Aalborg. To address the problem Aalborg Kommune would like to detect these incidents using Thermal Cameras. The origin of this master’s thesis is to carry out initial analysis on how to solve the problem using computer vision techniques. Three thermal cameras have been installed at the harbor. In this thesis an automatic surveillance system have been analyzed, developed and tested. From the thermal camera feed a person’s position is automatically extracted and tracked thought the scene using a Kalman filter. To prevent fatal accidents, the system is able to detect if a person falls into the water using a trip-wire and optical flow. The fall detector algorithm is able to detect 100% of all falls. The system only provides 0.08 false positive alarms per hour. Besides the fall detector a fall predictor has also been developed which will give an operator a warning before a person actually falls into the water. Test showed that the system is able to predict 23.67 % of the trip-wire activations beforehand. To develop the system a dataset of 155 hours of thermal video have been recorded during night hours. Persons trajectories have been annotated for 56 hours to use for training and module test – the remaining 99 hours have been used as an acceptance test.

The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.
Contents

Preface vi
Symbol- and Acronymlist 1

1 Introduction 3
1.1 Initial Problem Statement 5

2 Technical Analysis 7
2.1 Thermal Cameras 7
2.2 On-Land Scenario Analysis 10
2.3 In-Water Scenario Analysis 14
2.4 Detecting and Raising Alarms 16
2.4.1 Detect persons who fall into the water 17
2.4.2 Predict persons who are about to fall into the water 18
2.5 System Pipeline 19
2.6 Recap of Technical Analysis 21
2.7 Problem Statement 22

3 Project Overview 23
3.1 Project Delimitation 23
3.1.1 Time of interest 23
3.1.2 Weather invariance 23
3.1.3 Video Capture and Dataset 23
3.1.4 Handling and Ignoring Objects in the Scene 24
3.1.5 Position precision 24
3.1.6 Real-time processing 25
3.1.7 Abnormality Detection 25
3.2 Function Description 25
3.2.1 Actors 25
3.2.2 Use-cases 26
## Contents

3.3 Requirements ......................................................... 27  
3.4 System Design ..................................................... 28  
3.5 Dataset .............................................................. 29  

4 Person Extraction .................................................... 33  
4.1 Segmentation ......................................................... 33  
4.1.1 Design and Implementation ..................................... 37  
4.1.2 Evaluation ....................................................... 38  
4.2 Homography Mapping ............................................... 39  
4.2.1 Design ............................................................ 40  
4.2.2 Implementation and Evaluation ............................... 41  
4.3 Position extraction .................................................. 43  
4.3.1 Single person ..................................................... 44  
4.3.2 Wide-splitting of persons ...................................... 51  
4.3.3 Tall-splitting of persons ....................................... 61  
4.4 Module Test .......................................................... 67  
4.5 Recap ................................................................. 70  

5 Tracking ................................................................. 73  
5.1 Target Tracking ...................................................... 73  
5.1.1 Design ............................................................. 77  
5.1.2 Implementation .................................................. 86  
5.2 Track Management ................................................... 90  
5.2.1 Design ............................................................. 92  
5.2.2 Implementation .................................................. 95  
5.3 Module Test .......................................................... 96  
5.4 Recap ................................................................. 107  

6 Context Analysis ....................................................... 109  
6.1 Detecting and Predicting Falls .................................... 109  
6.1.1 Design ............................................................. 117  
6.1.2 Implementation .................................................. 123  
6.2 Test and recap ....................................................... 125  

7 Acceptance Testing .................................................... 127  
7.1 Fall Detection ....................................................... 127  
7.2 Fall Prediction ....................................................... 131  
7.2.1 Results ........................................................... 132
Preface

This report is written as the final project on the master’s programme Vision, Graphics and Interactive Systems at Aalborg University. The project has been carried out as a cooperation between Aalborg Kommune, Actas, Beredskabscenter Aalborg and Aalborg University.

The group would like to thank Kirsten Hein from the winter bathing club Lillebjørnen for using their facilities during the test performed at the harbor in the beginning of March. In addition to the test performed a thank to the rescue divers for providing safety during the test. A thank to Rasmus Krog from Aalborg Sportshøjskole for lending wetsuits to the test participants. Furthermore a sincere thanks to Henrik Svenstrup from Beredskabscenter Aalborg for providing equipment and access to the thermal cameras. Lastly a thank to Thomas B. Moeslund for guidance throughout the semester.

The code is implemented in **Python 2.7.6** and **OpenCV 2.4.11**

Aalborg University, June 2, 2016

Søren Bonderup  
sabb11@student.aau.dk

Jonas Olsson  
jolsso11@student.aau.dk
# Symbol- and Acronymlist

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOB</td>
<td>Binary Large Object</td>
</tr>
<tr>
<td>CD</td>
<td>Compact Disc</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complementary Metal Oxide Semiconductor</td>
</tr>
<tr>
<td>DEMA</td>
<td>Danish Emergency Management Agency</td>
</tr>
<tr>
<td>EM</td>
<td>Electromagnetic</td>
</tr>
<tr>
<td>FN</td>
<td>False Negative</td>
</tr>
<tr>
<td>FOV</td>
<td>Field Of View</td>
</tr>
<tr>
<td>FOV</td>
<td>Field-Of-View</td>
</tr>
<tr>
<td>FP</td>
<td>False Positive</td>
</tr>
<tr>
<td>K</td>
<td>Kelvin</td>
</tr>
<tr>
<td>LWIR</td>
<td>Long-Wavelength Infrared Radiation</td>
</tr>
<tr>
<td>MOTA</td>
<td>Multi Object Tracking Accuracy</td>
</tr>
<tr>
<td>MOTP</td>
<td>Multi Object Tracking Precision</td>
</tr>
<tr>
<td>MWIR</td>
<td>Mid-Wavelength Infrared Radiation</td>
</tr>
<tr>
<td>MWIR</td>
<td>Mid-wavelength IR</td>
</tr>
<tr>
<td>PTZ</td>
<td>Pan-Tilt-Zoom</td>
</tr>
<tr>
<td>PT</td>
<td>Pan-Tilt</td>
</tr>
<tr>
<td>RGB</td>
<td>Red, Green, Blue</td>
</tr>
<tr>
<td>TN</td>
<td>True Negative</td>
</tr>
<tr>
<td>TP</td>
<td>True Positive</td>
</tr>
<tr>
<td>VOx</td>
<td>Vanadium Oxide</td>
</tr>
</tbody>
</table>
1. Introduction

In the years between 2001 and 2013, 816 persons have died from unintentional drowning in Denmark. Of these, 199 drownings (24 %) happened at harbor areas, 8/10 cases involving alcohol beforehand [Syddansk Universitet, Statens Institut for Folkesundhed [33]]. One of these locations is in Aalborg, Denmark. Since 2008, 8 persons have drowned in Limfjorden, near Aalborg’s city centre [Nordjyske, Henrik Bo [27]].

![Figure 1.1: Number of unintential drowning deaths in Denmark between 2001-2013 (left). The location of 8 drowning deaths since 2008 in Limfjorden, Aalborg (right).](image)

Historically, only 16 % of the drowning incidents at harbors were witnessed by others [Syddansk Universitet, Statens Institut for Folkesundhed [33]]. Usually, the accidents are therefore only discovered when the person has drowned. To address the problem, Aalborg Kommune (Municipality of Aalborg) would like to detect these incidents using thermal cameras placed at strategic locations. The idea is to utilize each thermal camera’s ability to perceive heat signatures. Compared to regular RGB cameras, the advantage is that it also works during the night and being placed in a public setting, is not able to pick up a person’s identity.

Practically, three thermal cameras have been installed at Aalborg’s harbor front. Two static cameras are placed at the street Havnepromenaden, facing each other. The remaining camera, a PTZ camera, is placed at an adjacent bridge overlooking the entire harbor front. Thermal images from the cameras, including locations are shown in ![Figure 1.2](image)

The placement of cameras is chosen, as it is on a popular walking route, near the fjord. Additionally, it is also approx. 100 meters away from Jomfru Ane Gade, a popular “bar street” in Aalborg. It is often when having been in or near Jomfru Ane Gade, that persons disappear and drown in Limfjorden. Examples of drowning accidents include a 20 year-old in 2011 [Rohde-Brøndum [32]] and 21 year-old in 2015 [Nyhedsbureau [29]]. Both were last seen in Jomfru Ane Gade during night-time.

To carry out this project, Aalborg Kommune have started a collaboration with Beredskabscenter
Chapter 1. Introduction

Figure 1.2: Location of the cameras (upper left). Thermal images from 11 deg FOV PTZ (top right), 22 deg FOV static (bottom left) and 11 deg FOV static (bottom right). The part of Havnepromenaden of interest is 375 meters long, the entire northern side connected to the water. Here, approx. 145 meters is occupied by a harbor bath (Aalborg Havnebad) and a restaurant situated on an old ice breaker (Restaurent Elbjørn). This stretch is accessible by water, but obstructs the view from the PTZ camera.

Aalborg, which is a part of Danish Emergency Management Agency (DEMA) Nordjylland. For the past 4 years, Beredskabscenter Aalborg has had a 24h video surveillance center, surveying Aalborg Kommune’s properties and institutions. Using 300 cameras controlled by Milestone Systems’ Milestone XProtect®VMS system, operators are able to respond to alarms risen by the system. The idea is to detect the incidents when they happen. Ideally, the system should raise alarms for operators at Beredskabscenter Aalborg, which can then send response teams with access to boats to the location.

The time it takes before a rescue boat is ready for use in the water is usually between 4-5 minutes. Two major factors have an impact in survivability: the duration under water and the water’s temperature. Response time and an effective rescue is therefore a critical factor in survivability. The time between an accident takes place and is detected (by either an operator or bystander) must therefore be instantaneous. An idea is to investigate if the thermal cameras can detect the incidents before they happen.

A rich picture of the system is visualized in Figure 1.3

---

1DEMA Nordjylland is responsible for fire and rescue services for 580 000 people in the northern part of Denmark.
2Information from Henrik Svenstrup, Beredskabscenter Aalborg
1.1 Initial Problem Statement

An initial problem statement is composed, which forms the basis for the Technical Analysis, Chapter 2. How can a system consisting of thermal cameras be developed to send an alarm if people falls into the water?

Supplemental questions are added to the problem statement:

- Can thermal cameras be used to locate and track people if they are already in the water?
- Is it possible to give warnings before an accident takes place, to save costly response time?
- Is it possible to give alarms when an accident is taking place?

Figure 1.3: Illustration of the rich picture for the system. The system could be able to warn an operator about dangerous behavior from bypassers. Additionally, the system could raise an alarm when a person falls into the water. The operator can, based on the warning and alarms, track persons in the water.
2. Technical Analysis

This section will describe and analyze the technical aspects associated with the problems described in Chapter 1.

2.1 Thermal Cameras

The theory in this section is based on [Gade, Moeslund 10] unless otherwise specified.

Traditional (visible) cameras measure electromagnetic (EM) radiation, or photons, in wavelengths between 390 and 700 nm. This range is denoted the visible spectre, as it is in this range that the human vision can perceive photons. Traditional cameras work in a similar manner, by capturing photons emitted, or reflected by objects [Moeslund 25]. Since persons themselves do not emit photons in this spectre, the objects have to be illuminated by alternative light sources, such as the sun. This yields problems in environments where no light is available. This dependency for alternative light sources is therefore not ideal during all times of the day. It therefore proves impractical during nighttime, as ambient lighting is obviously not present (although can be compensated for by artificial lighting).

Thermal cameras on the other hand measure EM radiation in wavelengths between 0.7 and 1000 µm. In this range, denoted the thermal spectre, objects above 0 K emit radiation relative to its temperature, which can be measured. Different thermal sensors exist, and due to limitations in sensor spectrums, these are often divided into different categories [Opto Engineering 30]. Generally, two types of ranges are used w.r.t. persons, these are Mid-wavelength IR (MWIR, 3-8 µm) and Long-wavelength IR (LWIR, 8-15 µm) [Ghiass et al. 12]. MWIR and LWIR, referred to as thermal infrared, are able to measure direct emission from e.g. humans, as temperatures between 190-1000 K emit in this range. The radiation of an object can be found from Planck's wavelength distribution function. An example of emittance, calculated for different temperatures is shown in Figure 2.1.

Two factors control the emittance from an object: its temperature and emissivity, measured between $0 < \varepsilon \leq 1$ [30]. Perfect emitters, also called black bodies, have an $\varepsilon$ value equal to 1. Values closer to 0 do not emit radiation, and are called grey bodies. Relevant types of materials and their emissivity are shown in Table 2.1. Both the human skin and cloth have high $\varepsilon$ values. This means that a person's appearance in the camera is highly correlated with its temperature. The thermal radiation from body heat can therefore be utilized, such that persons can still be recognized, both during day- and nighttime. An example of images taken with both a visual and thermal camera at two times during the day is shown in Figure 2.2.

It is relevant to examine materials of objects, which could potentially have the same temperature as persons. If this case occurs, humans may become imperceptible as a cause of cluttering. In
Chapter 2. Technical Analysis

Spectral emittance for fixed temperature objects at different wavelengths

Figure 2.1: Thermal emittance for black body objects at different temperatures.

<table>
<thead>
<tr>
<th>Material</th>
<th>$\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human skin</td>
<td>0.98</td>
</tr>
<tr>
<td>Water</td>
<td>0.95-0.96</td>
</tr>
<tr>
<td>Asphalt</td>
<td>0.93</td>
</tr>
<tr>
<td>Cloth</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Table 2.1: Emmissivity for different types of materials. Higher $\epsilon$ values show better emittance.

Figure 2.2: Examples of images taken from a thermal (bottom) and a regular RGB camera (top) at 12:00 (noon, left) and 12 hours later at 00:00 (night, right).

[Kristoffersen et al. 20], the asphalt was warmer than most of the pedestrians as a cause of the weather being 20 °C sunny. The high $\epsilon$ value of asphalt could therefore prove problematic during the summer time, where the road and persons may have the same temperature. The contrast between person and background might therefore be less visible. An example is seen in Figure 2.3.

Besides the asphalt making cluttering when it gets warmer water puddles can also cause troubles. As described in [Gade et al. 9] glossy surfaces will make a reflection of the person. A thing to have in mind is that pavement reflections will always point against the camera, since the radiation is emitted by the persons. Therefore when persons walk close to glossy areas on the pavement, the reflection will always be a more or less vertical mirrored against the camera as described in illustration Figure 2.4.
2.1. Thermal Cameras

Figure 2.3: Image from a thermal camera (left) and a visible camera (right). The ambient temperature is 12 °C (sunny), which causes the asphalt to have the same temperature as human clothing. The contrast between person and asphalt is therefore at a minimum.

Figure 2.4: Showing how reflections are emitted by a person itself. Left figure shows how a person will emit its reflections by a glossy area on the pavement (blue) pointing towards the camera. Right figure shows how water puddles on the pavement causes a reflection from the person and to the camera.

The cameras used in this project are provided by Hikvision[1]. Each camera includes both a thermal and visual (RGB) sensor. Specifications for the cameras are shown in table 2.2.

<table>
<thead>
<tr>
<th></th>
<th>DS-2TD4035D-50</th>
<th>DS-2TD2235D-25</th>
<th>DS-2TD2235D-25</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PT capable:</td>
<td>Pan: 360° endless; tilt: -15° ↔ 90°</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Zoom:</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Codec:</td>
<td>H.264/MJPEG/MPEG4</td>
<td>Fixed</td>
<td>Fixed</td>
</tr>
<tr>
<td><strong>Thermal stream:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Type:</td>
<td>VOx Uncooled focal plane detector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resolution:</td>
<td>384 × 288 pixels</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Framerate:</td>
<td>25/30 fps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lens:</td>
<td>25 mm</td>
<td>50 mm</td>
<td>25 mm</td>
</tr>
<tr>
<td>FOV:</td>
<td>21.7°</td>
<td>11°</td>
<td>21.7°</td>
</tr>
<tr>
<td><strong>Visual (RGB) stream:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Type:</td>
<td>1/1.8&quot; progressive scan CMOS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resolution:</td>
<td>1920 x 1080 pixels</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Framerate:</td>
<td>25/30 fps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lens:</td>
<td>4.3 - 129mm</td>
<td>40mm</td>
<td>23mm</td>
</tr>
<tr>
<td>FOV:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extras:</td>
<td>High-performance IR arrays (up to 150m)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auto focus</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2: Specifications for the three cameras used in the setup[14] [15].

Chapter 2. Technical Analysis

Hikvision estimate their camera’s range according to Johnson’s criteria. Johnson’s criteria is a measure, which describes an observer’s possibility of determining objects on the image sensor. The criteria gives a human observer a 50% chance of determining the target person at three different stages. Assuming a person is 0.5 m. wide, the stages are:

1. **Detection**: An object is present. 2 pixels/0.5 m.
2. **Recognition**: The object can be distinguished between other classes (e.g. person vs. car). 8 pixels/0.5 m.
3. **Identification**: The object’s characteristics can be determined (e.g. man vs woman). 13 pixels/0.5 m.

![Figure 2.5: Image of persons at detection (left), recognition (middle) and identification (right) range.](image)

In this project, it would be desirable to recognize pedestrians. From the data sheet describing the cameras [Hikvision], the maximum distance from the two cameras where it should be possible to recognize humans are stated as:

- DS-2TD4035D-25 (25 mm): 125 meters
- DS-2TD4035D-50 (50 mm): 200 meters

This means that the system should be able to detect persons at a distance of 125 m. for the 25 mm. camera and a distance of 250 m. for the 50 mm. camera. Persons at this range are approx. 8 pixels wide and 24 pixels in height.

### 2.2 On-Land Scenario Analysis

Due to the placement of the cameras, different types of objects occur in the scene. Of these objects, only those that could potentially fall in the water are of interest. In order to define objects of interest, the objects must be 1) detectable by the IR camera, therefore contain heat visible to the IR camera and 2) contain movement, which could potentially cause it to fall into the water. Examples of objects of interest are shown in [Figure 2.6] and in this frame five different events appear:

1. A person walking with a child
2.2. On-Land Scenario Analysis

2. A person walking with a baby carriage
3. A person walking alone
4. A person biking
5. A car

Figure 2.6: Several different events in same scene, as seen with the RGB (left) and thermal camera (right): A person with a baby carriage (red arrow), an adult walking hand-in-hand with a child (orange arrow), a person biking (green arrow), a person walking (blue-arrow) and a person in a car (pink arrow).

Two contexts are interesting w.r.t. to objects. Firstly, which types of objects and in which scenario they occur in the scene. Secondly, which patterns and trajectories the objects occur in. In order to identify these contexts, it is decided to analyze videos on a Wednesday and a Saturday from 15 to 15. These represent both a day in the mid-week and weekend in the spring of 2016. Information about the videos are shown in Table 2.3.

<table>
<thead>
<tr>
<th></th>
<th>Wednesday</th>
<th>Saturday</th>
<th>Wednesday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>March 9</td>
<td>April 9</td>
<td>April 6</td>
</tr>
<tr>
<td>Temperature</td>
<td>2°C</td>
<td>0°C</td>
<td>7°C</td>
</tr>
<tr>
<td>Wind</td>
<td>4 m/s</td>
<td>3 m/s</td>
<td>5 m/s</td>
</tr>
<tr>
<td>Rain</td>
<td>0 mm</td>
<td>0 mm</td>
<td>10 mm</td>
</tr>
</tbody>
</table>

Table 2.3: Weather information from the two videos used to annotate objects of interest and their behavior.

Object types and scenarios

As written above different types of objects (events) will occur in the scene. These events might also be effected by either partial - or full occlusion.

- Partial occlusion: where only parts of a person can be seen. This means that a position should either be estimated based on the remaining part of the person or at some point removed due to the inaccuracy of making a proper estimate. In Figure 2.7 the person is partial occluded by the pole and a few frames later, total occlusion occurs.
- Full occlusion: where it is not possible to see a person. Therefore it is not possible to find a position.
Chapter 2. Technical Analysis

Figure 2.7: Partial and total occlusion will often occur in the scene of camera 1, due to the pole which is placed in the entire center of the FOV.

In Figure 2.8 examples of different events are shown.

Figure 2.8: Samples of different types of persons / events appearing in the scene. From left to right: Single person, person with dog, couple cuddling, person on a bike, car.

Initial video browsing shows that the three most occurring events are: persons, persons on bikes and cars. To get a quantitative measure of the frequency events occur, statistics of these events have been noted and are shown in Figure 2.9.

Figure 2.9: Bar-plot showing the number of appearances of persons, person biking [bikes] and cars. In the hours between 23 and 07 the “traffic” is rather limited. In the remaining hours of the day is rather populated – 100+ events. (the statistics calculated has been recorded for events appearing in camera 2)

\(^2\)The videos can be found in a folder on the CD under analysis/scenario_analysis/.
2.2. On-Land Scenario Analysis

From the bar-plot [Figure 2.9], it is obvious that the frequency of people (events) passing the scene is lowest in the time interval from 23-07 (which from now on will be referred to as night hours). The average time between each events is:

- **Day-hours (week days):** In average 142 events / per hour → 25.36 seconds between each event.
- **Day-hours (weekend):** In average 261 events / per hour → 13.79 seconds between each event
- **Night-hours (week days):** In average 11 events / per hour → 333 seconds (5 min 33 seconds) between each event.
- **Night-hours (weekend):** In average 43 events / per hour → 83.72 seconds (1 min 20.72 seconds) between each event

Since most drowning accidents happen without supervision (Chapter 1), it should be considered to focus on the night hours – since this is the time of the day where frequency of people appearing is the lowest.

In the next section behavior of the people passing the scene will be analyzed.

**Object patterns and trajectories**

To analyze people’s behavior, the Wednesday has been manually annotated. The annotation has been carried out for persons passing the scene – this does not include people biking. Due to the high frequency of people occurring in the scene during day-time, the annotation has been carried out in the time interval from 15-17 and at night from 23-07. By plotting these positions a histogram is constructed for each pixel, resulting in different colors representing the number of times a person is walking at this position in the image.

![Figure 2.10: Heatmap of trajectories in each pixel for 232 persons during the day (left) and for 44 persons during the night (right). The videos are captured on March 9, 2015.](image)
Figure 2.11: Heatmap of trajectories in each pixel for 237 persons during the day (left) and for 42 persons during the night (right). The videos are captured on April 6, 2015.

From the two images in Figure 2.10-2.11 it seems that people does not walk as close to the harbor’s edge, as they do in day-hours. In day hours people use the entire harbor.

Work in trajectory analysis is often based on, that person’s trajectories are determined beforehand [Pellegrini et al.,31]. This indicates that persons prefer direct paths to their goals [Kitani et al.,19] for instance on pavements thus avoiding any obstacles in their way. A plausible hypothesis is, that during nighttime persons use the harbor with the goal of transportation from A to B, whereas persons during the daytime may have recreational incentives, thus walking closer to the harbor front. This is supported from visual inspection, where persons are observed using the harbor’s facilities, such as benches, which imply relaxation.

### 2.3 In-Water Scenario Analysis

To investigate the second part of the initial problem statement Section 1.1 which concerns the idea of using a thermal camera to search for people who is in the water, a preliminary test is conducted to test if recognition in the water is possible. The test has been conducted in the harbor bath (using the winter bathing club Lillebjørnen’s facilities). Two scenarios were investigated where 1) people were swimming normally in the water and 2) were simulating panic (feverish splashing, acting as about to drown). Examples of the two scenarios are shown in Figure 2.12.

<table>
<thead>
<tr>
<th></th>
<th>Wednesday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date:</td>
<td>March 3</td>
</tr>
<tr>
<td>Temperature:</td>
<td>5°</td>
</tr>
<tr>
<td>Water:</td>
<td>3°</td>
</tr>
<tr>
<td>Wind:</td>
<td>3m/s (*)</td>
</tr>
</tbody>
</table>

Table 2.4: Weather information for the day where the test was conducted. (*) No actual wind due to sheltered condition in harbor bath.
2.3. In-Water Scenario Analysis

![Figure 2.12: Sample images from a preliminary test in the harbor bath in Aalborg, where left image shows a person simulating an unintentional drowning, and right image shows a person swimming in the water.](image)

One of the participants were told to swim, duck the head under the water, and continue swimming (right image Figure 2.12). It takes approx. 1-3 seconds before the head is warm enough to be visible by the thermal camera, and as shown in Figure 2.12 the only part of the body which is visible from the thermal camera is the head. When water was splashed/covering the head, the heat radiation is not visible by the thermal camera. Another thing to have in mind is, that the head/neck of a body is only ~7.30% of the human body (Tözeren 37), therefore it is only a small amount of the human body which can be captured, when a person is in the water.

To validate the above the motion has also been analyzed. To analyze the motion the dense optical flow is plotted in the test, carried out at the harbor. In Figure 2.13 examples from the analysis are shown.
From a visual inspection of the examples in Figure 2.13 it seems to be difficult to search for people in the water. Waves and reflections in the water might cause variations in the captured thermal video, which might cause it to be difficult to separate between a person in the water and noise.

### 2.4 Detecting and Raising Alarms

When dangerous behavior or persons falling into the water, an operator should be informed. Two terms are used in terms of raising alarms: *Positives* which are persons falling into the water and *negatives*, which are persons walking (normally) on land. Alarms risen at operators can either be identified as correct (true) or incorrect (false) depending on the state of the person. This forms the confusion matrix, containing different states for the system in Table 2.5.

<table>
<thead>
<tr>
<th>State</th>
<th>Alarm</th>
<th>No Alarm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accident occurring</td>
<td>True Positive</td>
<td>False Negative</td>
</tr>
<tr>
<td>No accident occurring</td>
<td>False Positive</td>
<td>True Negative</td>
</tr>
</tbody>
</table>

Table 2.5: Confusion matrix classifying actual states persons might have versus outputs of the system.
2.4. Detecting and Raising Alarms

Typically, designing a system is a trade-off between having false negative (FN) or false positive (FP) errors. In the scope of this project, FN errors could prove fatal, as persons may drown if no alarms are risen. Rather, it would be preferred to raise false positive (FP) alarms at the operator, which in worst-case distracts the operator. An excessive amount of FP errors may have an impact on the system’s credibility, causing operators to ignore any warnings risen by the system. Studies for traditional CCTV applications show that the majority of operators are only able to effectively monitor 4 streams at a time \[\text{Wallace, Diffley}^{[40]}\]. The system should therefore aim at minimizing the FP errors, while causing no FN errors.

From the Section 1.1 it is desired to design a system which can send an alarm if persons fall into the water. To detect such a situation, different approaches can be taken:

1. *Predict* persons who are *about* to fall into the water
2. *Detect* persons who fall into the water

A more detailed description of item 1-2 including when to raise alarms will be described in the following.

2.4.1 Detect persons who fall into the water

Two approaches are considered when detecting if persons fall into the water:

- Detect person’s crossing the harbor’s edge
- Detect persons during the fall

Many cameras nowadays, including the 3 thermal cameras used \[\text{Section 2.1}\] have built-in detection systems for cross line detection (also known as trip-wire detection). These detection systems work by a predefined virtual line in the camera’s FOV. In case the predefined line is intersected, an alarm will be raised and objects can be detected.

![Figure 2.14](image)

*Figure 2.14: The three areas around the harbor’s edge (white): pavement area (green), raised anchoring area (red) and water area (blue). An aerial view is seen (left) with overlaid area (right) as viewed from camera 1.*

The area around the harbor’s edge is divided into three areas: pavement area, raised area where ships may anchor and water. Obviously, if the system should predict or warn beforehand, persons cannot be positioned in the water area. As seen in *Figure 2.10* a person’s normal trajectory may also be the pavement area close to the edge. The risk area is therefore assessed

\[^{34}\text{monitor streams are also most optimal at Beredskabscenter Aalborg (ref. Henrik Svenstrup)}\]
to be the raised anchoring area, approximately 60 cm. Figure 2.14 shows the different areas, as marked by camera 1.

During the analysis of people in the water Section 2.3, it was observed that when a person falls into the water, a rather high area is affected by a change in brightness (area around the person during fall and by the water’s splash). In order to see if this visual observation was capture-able by the thermal camera, the motion in the scene is analyzed using optical flow. In Figure 2.15 two images are shown. The left shows the optical flow during the fall and the right where the person disappears in the water. From this example it is obvious that there is a rather high motion field around the person during the fall.

![Figure 2.15: 2 frames where the optical flow is plotted. Left: frame during the fall. Right: frame during disappearing.](image)

From this it seems possible to use the optical flow, for detecting if a person is falling. As described in Section 2.3 motion may also occur from waves in the water, so a more thorough analysis needs to be carried out. A “falling accident” would ideally cause a more dominant motion flow in the vertical direction due to gravity, which might be possible to use to detect falling persons.

### 2.4.2 Predict persons who are about to fall into the water

As stated in Chapter 1 time is a critical factor in ensuring survivability. Therefore, it would be beneficial to raise a warning if there is a potential risk of a person falling into the water. The task generally involves determining and detecting abnormal behavior, which is divided into two approaches [Laxhammar 22].

- **Signature-based** detection, where knowledge of specific behavior is predefined in existing templates or models. An example of signature-based detection would be to utilize existing knowledge of the harbor’s environment, for instance that persons heading towards the trip-wire line may exhibit dangerous behavior. Signature-based methods may not be sufficient or realizable, since persons may exhibit different, dangerous behaviors which are not always known beforehand. As a supplement, it can be mentioned that no known records exist of persons falling into the water at the harbor’s areas.

- **Outlier-based** detection, which is based on knowledge of existing, normal behavior. Typically, this is defined from knowledge of persons trajectories over time, for instance as shown in Figure 2.10. Similar contexts, such as pedestrian detection and analysis also combine these trajectories with known formulas for social interaction [Pellegrini et al. 31]. Detection
can either be evaluated *online* frame-by-frame or *offline* after each trajectory is fully formed. Within the scope of predicting persons about to fall into the water an offline approach would not suffice as warnings would be raised *after* an accident has occurred. Rather, an online approach is preferred.

2.5 System Pipeline

In order to process the raw thermal feed into information, such as alarms and warnings, several steps are needed. This is done using a pipeline as shown in Figure 2.16.

![Figure 2.16: Video pipeline as presented in [Maggio, Cavallaro] and adjusted for the scope of this project. The pipeline consists of several steps, from inputted image to outputting alarms for the operator.](image)

The first two steps, *person extraction* and *tracking*, is a general part of a video pipeline as described in [Maggio, Cavallaro]. For the scope of this project, the *context analysis* is added to the pipeline, whose task is to determine an output for the system. The pipeline and its parts are described in the following.

**Person extraction**

<table>
<thead>
<tr>
<th>In</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video frame, $I_k$</td>
<td>Position for object(s), $Z_k$</td>
</tr>
</tbody>
</table>

The first step in the pipeline is to detect person’s positions, $Z_k$ in each frame, $I_k$. The *Object extraction* separate foreground (the persons) from the background (the scene), known as segmentation. This can be done in one of two ways: through use of an object model, or through background modeling [Maggio, Cavallaro]. The former is usually a detector, which is able to find pre-defined models, or appearances of the object of interest. Typically this includes training a classifier for detection on a pre-defined set of features. An example is [Viola et al.], which detects pedestrians using rectangular filters, trained using a cascaded AdaBoost classifier. The latter method, a background model approach, uses a classifier for a known background to find foreground pixels. This can for example be done by modeling each pixel’s distribution. An example is using Gaussian Mixture Models for each pixel, as in [Zivkovic]. The background modeling approach works well with static camera’s, as the only movement in the camera is formed by moving objects, such as pedestrians.

The first task for the *object representation* is to find a general representation for the found foreground objects. This can be done in various ways, such as articulated joints, or simply rectangular bounding boxes around the Binary Larger OBjects (BLOB). The choice of representation is a trade-off between having invariance vs accurately describing the object [Maggio, Cavallaro]. An example of this is seen in Figure 2.17. In the scope of this project, it would be better to choose a less accurate representation, such as a bounding box, rather than an incomplete, complex notation.
Figure 2.17: Problem associated with a more complex, but accurate object description, such as using articular joints. The pedestrians are often occluded, which may cause problems in calculating the joints. Another approach is to create a less accurate, though more robust object descriptor, such as a bounding box.

The last task of the Target representation is to find a representation for the object’s position within the scene. Ideally, when the position of each person in the scene is known, the position can be used to model a person’s behavior as written in Section 2.4.2. Since this project contains several cameras, it would be beneficial to have a generalized representation of the objects positions. This can be done through camera calibration, where the positions on the image plane can be translated to world coordinates. To make this transformation, a homography mapping of the cameras coordinates must be made into world space. An illustration of mapping between the two cameras into a general plane, can be seen in Figure 2.18.

Figure 2.18: Illustration of principle for a homography. The cameras extrinsic parameters are estimated, such that positions on the image plane can be projected onto the (world) reference plane (left). The reference positions for persons must be in the same location (on the object), as it would otherwise give wrong positions (right).

The module must output a position for an object on the image plane, in world coordinates. This requires defining the object’s reference position. Since the homography is done through a monocular camera, the image coordinates are assumed to be placed directly onto the mapped plane. In this case, tracking person’s faces is not optimal, as persons have varying heights. This causes problems, as shown in Figure 2.18. Using the homography method, the reference point must be as close to the mapped plane as possible. Therefore, the reference point must be near the ground, e.g. the object’s feet.
2.6 Recap of Technical Analysis

Tracking

<table>
<thead>
<tr>
<th>In</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position for object(s), $Z_k$</td>
<td>Tracked targets, $X_k$</td>
</tr>
</tbody>
</table>

The second step in the pipeline is to assign positions, $Z_k$, to tracked targets, $X_k$. When a position is incoming, it is assigned to a tracker. This is known as target tracking. The tracker handles the trajectories of targets and determine actions, if:

- the object is occluded and therefore can’t be found in the scene, thus the positions must be estimated
- the object leaves the scene, known as “target death”

The incoming spatial positions are discrete and are therefore not associated with a target over time. This should be handled, such that incoming positions are assigned to existing trackers, adding temporal information. This process is known as track management. The track management should also handle the creation of new trackers, based on objects entering the scene. This “target birth” can occur in a number of ways [Maggio, Cavallaro][24]:

- Persons incoming from the image’s edges
- Persons becoming visible for the person extraction, as they approach camera.
- Persons appearing from other objects, such as cars or boats

Context analysis

<table>
<thead>
<tr>
<th>In</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracked targets, $X_k$</td>
<td>Operator output</td>
</tr>
</tbody>
</table>

The last step in the pipeline is to process the target trajectories, $X_k$, such that an operator can be warned or alarmed by incoming events. As described in Section 2.4, this can be done by:

- Triggering a warning if a person is about to fall into the water.
- Raising an alarm if a person is falling.

2.6 Recap of Technical Analysis

Chapter 2 describes the problems associated with developing a system for notifying an operator with either alarms (person falling) or warnings (dangerous behavior).

The thermal cameras used in this setup measure radiation from object with temperatures between 190 to 1000 K (-83 to 727 °C). The thermal cameras are therefore able to pick up thermal body radiation. The emissivity of human skin and cloth are very high, and can thus be used to recognize persons at all times of the day. Despite this it is important to have in mind that reflections from glossy surfaces can occur, together with the problem of cluttering if the heat signatures between persons and pavement are uniform.
An on-land analysis identified objects of interest, which needs to be handled in the system. This is objects which contain motion and are detectable by the camera. From the annotated video, it can be seen that persons appear in several settings like person on bikes, person holding hands (1 object contains two persons), partial and total occlusion etc. The trajectories that these persons take, form a pattern. This pattern could potentially be used to detect if a person exhibit dangerous behavior by either signature- or outlier-based methods. This analysis also proved that the risk of falling into the water, without being spotted, is highest during night hours between 23 – 07.

A preliminary test was made in Aalborg’s harbor bath. An in-water analysis of the subjects, showed that detecting persons in-water can be very difficult. This is a result of two things: The persons being covered in water and is thus almost invisible to the camera and waves causing disturbances in the scene. Despite the difficulty of detecting people in the water, it was observed that during a person’s fall from the harbor edge and into the water, detectable motion occurs.

A pipeline was introduced, which forms a basis for a general framework. The pipeline consists of several steps. In the person extraction step, the inputted video must be segmented to distinguish objects from background. From the found objects, a representation must be made. The representation should be adequately robust, in terms of invariance, such that occlusions and changes in pose can be handled. The outputted positions must be assigned to new or existing targets, known to be in the scene. This add temporal information to the discrete positions, such that target’s trajectories can be identified.

As described in [Chapter 1], the system should be able to give warnings before and alarms when accidents occurs. The former can be done using a trip-wire, defined near the harbor edge. The latter can be done in multiple ways. Examples include those based on object’s features (e.g. speed) and modeling object’s trajectories. This compose the last part of the pipeline, which analyzes targets trajectories and outputs alarms or warnings if necessary.

In order for the system not to be un-reliable it is important to have a low rate of false positives (alarms where people are not falling), but also false negatives can absolutely not be tolerated – since can lead to drowning.

### 2.7 Problem Statement

Based on the analysis of the initial problem statement Section 1.1 and the recap of the technical analysis Section 2.6 the following problem statement is formulated:

How can a system be developed to detect and track persons on the harbor, and warn an operator if the trajectory is abnormal or raise an alarm if a person fall into the water?
3. Project Overview

Based on the Introduction, Chapter 2 and Technical Analysis, a project overview will be described. This includes the project decimations, function description and how the system is designed. Lastly the dataset will be described.

3.1 Project Delimitation

The developed system and constructed dataset will comply with the following delimitations described in this section.

3.1.1 Time of interest

In this subsection, an analysis was carried out showing that the most critical time to fall into the water will be during night hours. This is due to the frequency of persons walking at the harbor is very low. With respect to this, the time of interest (in the scope of people drowning) is chosen to be night hours from 23 – 07.

3.1.2 Weather invariance

As described in Section 2.1, examples show that the asphalt may cause cluttering, which makes persons imperceptible when walking on asphalt pavement. It is chosen not to handle this type of cluttering, since the time interval of interest is from 23 – 07 and cluttering caused by warm weather (person and pavement same temperature) are assumed not to occur often. Besides cluttering the pavement can also cause reflections – since these reflections often can occur eg. in rainy weather, the system should be able to handle reflections.

3.1.3 Video Capture and Dataset

To capture video from the harbor, the two static cameras (Hikvision DS-2TD2235D-25) described in Section 2.1 will be used. These cameras are able to capture both RGB and thermal video. The system will be based on the thermal video, since this will make the system able to see no matter the time of the day (day vs. night). Furthermore privacy issues are avoided.

The system will be developed and tested on camera 1 (camera placed at the bridge). From camera 1 the edge of the harbor is free of sight (including water) up to the distance where the system is going to be functioning – compared to camera 2 where the ship (restaurant) and
.bridge onto it, covers much of the harbor edge. Furthermore the double amount of time used for annotating would be needed.

A dataset is needed in order to develop and test the system. The dataset should represent hours of interest which is night hours from 23 – 07, including both week and weekend days. The dataset should include training data, test data for module tests and data for acceptance tests.

3.1.4 Handling and Ignoring Objects in the Scene

The project will be focusing on handling persons in the scene, therefore noise like animals (birds, dogs etc.) should be ignored. Since the focus is on night hours it is assumed that children (if present) will always be walking together with an adult.

From the plot in subsubsection 2.2 showing the different events with cars will not occur very often, therefore cars will not be handled in the project.

3.1.5 Position precision:

Due to the expected ranges the system should be able to extract a person’s position in the area up to 125 meters away from camera 1. The accuracy of the tracking part of the system depends on the position precision of each person.

As a quantitative requirement the precision which is obtained from the segmentation part, should not vary more than 10 % of the person’s height (~2 pixel). Translated into world coordinates, this would yield different values depending on a person’s placement relative to the camera and thus in the image’s FOV. Measured from column 105 in the image the pixel size in meters for each row is shown in Figure 3.1.

![Figure 3.1: Showing the vertical pixel size (in meter) to row correspondence.](image)

From measurements, the lowest image row where persons enter are 125 meters away from the camera corresponding to row 76. The row where persons are closest to the camera is row 287 (bottom border). Three different areas are defined within the image’s FOV based on the pixel size in the image’s rows listed in Table 3.1.

---

1Training and test data is also recorded for camera 2 for future use, although not annotated and used.

2This column is from visually inspection the longest possible distance for a person to appear at row 76.
### Table 3.1: Three different areas are defined to make a requirement for the pixel accuracy at different distances from the camera.

<table>
<thead>
<tr>
<th>Zone</th>
<th>Rows</th>
<th>Max Pixelsize [m./pixel]</th>
<th>Accuracy [m.]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>76-108</td>
<td>2.83</td>
<td>5.66</td>
</tr>
<tr>
<td>2</td>
<td>109-141</td>
<td>1.00</td>
<td>2.00</td>
</tr>
<tr>
<td>3</td>
<td>142-287</td>
<td>0.50</td>
<td>1.00</td>
</tr>
</tbody>
</table>

In Figure 3.2 the three areas are marked with red (zone 1), green (zone 2) and blue (zone 3).

![Figure 3.2: Showing the three zones defined in Table 3.1, where red (zone 1), green (zone 2) and blue (zone 3).](image)

#### 3.1.6 Real-time processing

The system is intended to raise an alarm if a person falls into the water. As described in Chapter 1, it usually takes between 4-5 minutes before a boat can be in the water – in order not to further delay the rescue team, real-time processing of the video is needed.

#### 3.1.7 Abnormality Detection

To analyze and implement abnormality detection a working framework for finding trajectories is needed. Furthermore lots of data needs to be processed and analyzed. Therefore the abnormality detection method will be based on the velocity and direction of a trajectory. Hereby a proof-of-concept will be implemented and tested, but also make it possible for future investigation of abnormality detection.

### 3.2 Function Description

For the reader to have an overall understanding of the system, a use-case diagram is made. In Figure 3.3 the different actors and use-cases in the system.

#### 3.2.1 Actors:

In Table 3.3 the actors are shown, together with an individual description:
3.2.2 Use-cases:

In Table 3.2, the use-cases are shown, together with an individual description:

<table>
<thead>
<tr>
<th>Use-cases</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detect person</td>
<td>The system should be able to detect persons in the incoming video from the thermal cameras. Positions are found by individually segmenting the frames and hereby discarding objects considered as noise.</td>
</tr>
<tr>
<td>Track person</td>
<td>When the positions of each object is found, the targets will be tracked during the video sequence.</td>
</tr>
<tr>
<td>Detect fall</td>
<td>Using the position of a person the system should detect if a person falls into the water.</td>
</tr>
<tr>
<td>Predict future fall</td>
<td>Based on the current trajectory of the person, the system should predict if the person will fall into the water.</td>
</tr>
<tr>
<td>Warning/ Alarm</td>
<td>An operator should be notified in case a person are about to or falls into the water.</td>
</tr>
</tbody>
</table>

Table 3.3: Description for each use-case in the system.
3.3 Requirements

In Chapter 1 Introduction, Chapter 2 Technical Analysis Section 3.2 Function Description and Section 3.1 Project Delimitation different aspects of designing the system have been described. In accordance with this the following overall requirements are set and divided into modules:

<table>
<thead>
<tr>
<th>Description</th>
<th>Specification</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Processing time</td>
<td>Must be able to process in real-time (processing time must not surpass video length)</td>
<td>3.1.6 p. 25</td>
</tr>
</tbody>
</table>

Table 3.4: Requirements with respect to the overall system.

<table>
<thead>
<tr>
<th>Description</th>
<th>Specification</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>2) Person extraction</td>
<td>Must be able to detect and recognize individual persons</td>
<td>3.1.4 p. 24</td>
</tr>
<tr>
<td>3) Position representation</td>
<td>Must be able to represent a person’s position on the harbor’s pavement plane in world coordinates</td>
<td>2.5 p. 20</td>
</tr>
<tr>
<td>4) Position precision</td>
<td>Must be able to estimate a person’s position with an accuracy of 5.66 m. in area 1, 2.00 m. in area 2 and 1.00 m. in area 3</td>
<td>3.1.5 p. 24</td>
</tr>
<tr>
<td>5) Full occlusion handling</td>
<td>Must be able to predict person’s positions when full occlusion is present</td>
<td>2.5 p. 21</td>
</tr>
<tr>
<td>6) Partial occlusion handling</td>
<td>Must be able to extract a person’s position when partial occlusion is present</td>
<td>2.2 p. 12</td>
</tr>
<tr>
<td>7) Temporal tracking</td>
<td>Must add temporal information to discrete, spatial information</td>
<td>3.4 p. 21</td>
</tr>
<tr>
<td>8) Segmentation</td>
<td>Must segment persons from the background</td>
<td>2.5 p. 19</td>
</tr>
<tr>
<td>9) Weather invariance</td>
<td>Object representation and position should be invariant to reflections caused by rain on the pavement</td>
<td>2.1 p. 7, 3.1.2 p. 23</td>
</tr>
</tbody>
</table>

Table 3.5: Requirements with respect to the person extraction and tracking part.

<table>
<thead>
<tr>
<th>Description</th>
<th>Specification</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>10) Motion detection</td>
<td>The system should be able to detect when people fall into the water</td>
<td>2.4 p. 16</td>
</tr>
<tr>
<td>11) Alarm rate</td>
<td>The number of FN errors must be 0</td>
<td>2.4 p. 16</td>
</tr>
<tr>
<td>12) Alarm rate</td>
<td>The number of FP errors should be at a minimum</td>
<td>2.4 p. 16</td>
</tr>
<tr>
<td>13) Warning distance</td>
<td>Must give warnings 5 seconds before persons are on collision course with the harbor’s edge</td>
<td>3.1.7 p. 25</td>
</tr>
</tbody>
</table>

Table 3.6: Requirements with respect to the context analysis.
Chapter 3. Project Overview

### Dataset

<table>
<thead>
<tr>
<th>Description</th>
<th>Specification</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>14) Recording</td>
<td>Videos of the thermal stream should be captured between 23:00-07:00</td>
<td>2.2 3.1.3</td>
</tr>
</tbody>
</table>

Table 3.7: Requirements with respect to the dataset.

## 3.4 System Design

The overall design of the system is shown in Figure 3.4, which is based on the Function description Section 3.2. The system has been split into 3 general modules Person extraction, Tracking, Alarm management. These 3 modules have their origin from the System Pipeline Figure 2.16. Each general module is further split into sub-modules. Below each sub-module a short description is added.

![System Design Diagram]

Figure 3.4: Describing the overall system design. The system is split into 3 general modules; Person extraction, Tracking and Alarm management. Each of these general modules are for easier understanding split into sub modules – in this design.

The initial step is to acquire the thermal video from the cameras. When the video is available each frame will be handled individually. First submodule module Object extraction will start by performing a Segmentation to separate foreground from the background. After segmentation BLOB extraction finds all the moving objects in the scene. Second part of the first general module is to represent the object Object representation. An object will be represented by a bounding box. Having the bounding box a reference point will be found – and mapped into world coordinates Mapping to world coordinates. The output from the first general module will be the world coordinate position for the objects, symbolized with the database symbol Object positions.

Next part of the system design Tracking, is to handle the object positions and track them over time. The position are first assigned to a tracker Assign position to tracker. After having the positions assigned to a tracker, the sub-module Target tracking will update and predict information about trackers and their positions. The information from the target tracking will be send back to Track management, which will output information about the trackers. This information will be id, position, acceleration, velocity and timestamp. Using the output from the general module Tracking, information will be input to the module for Alarm management.
3.5 Dataset

Last module of the system design is *Alarm management*, where a fall detector and fall predictor will run in parallel. If *fall predictor* stats a person is about to fall into the water a **warning** will be raised. Otherwise if a person is detector as *falling* from the harbor edge and into the water an **alarm** will be raised.

### Naming Convention

The videos are named according to the following format `cam_X_YY-MM-DD-hh_mm_ss.mkv`, where *X* is the camera identifier (1 or 2), *YY* is the year (16), *MM* is the month (01-12), *DD* is the date (01-31), *hh* is the hour at recording start (00-23), *mm* is the minute at recording start (00) and *ss* is the second at recording start (00). An example is `cam_1_16-03-20_23_00_00.mkv`.

As described in **Subsection 3.1.3** the focus is on night hours from 23:00-07:00, divided into videos of 1 hour each.

### Annotation Tool

An annotation tool has been developed in Python/PyQt4, which is able to annotate person and their positions in each frame. The positions are linearly interpolated for each frame between annotated via points.

![Annotation tool](image)

**Figure 3.5:** Annotation tool showing number of trackers (left) and positions for the chosen tracker (right). In each frame, the via points are shown as red dots and the interpolated positions are shown as blue lines.

For each video, an output file is generated. The format is `tracker id, x, y, frame nr`. A sample output of two annotated persons are shown in [Figure 3.6](#).

The tool can also be used to keep track of the events occurring, and can be saved separately using the “Save stats”.
Figure 3.6: Example of outputted txt file containing positions for each annotated person in the format tracker id, x, y, frame nr.

Dataset for Module Development and Testing

The dataset consists of IR-videos from week 12: 21/3-27/3 2016, the “Easter holidays”. In total 56 hours of data has been annotated. The division of the dataset into training/ test data used in each module is shown in Table 3.8.

<table>
<thead>
<tr>
<th>Day</th>
<th>YY-MM-DD</th>
<th>Type</th>
<th>Temperature*</th>
<th>Rain</th>
<th>Wind</th>
<th>Persons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday</td>
<td>16-03-21(20)</td>
<td>Training</td>
<td>1 °C</td>
<td>&gt;1 mm.</td>
<td>7 m/s</td>
<td>50</td>
</tr>
<tr>
<td>Tuesday</td>
<td>16-03-22(21)</td>
<td>Test</td>
<td>-2 °C</td>
<td>&gt;1 mm.</td>
<td>4 m/s</td>
<td>67</td>
</tr>
<tr>
<td>Wednesday</td>
<td>16-03-23(22)</td>
<td>Training</td>
<td>-4 °C</td>
<td>0 mm.</td>
<td>5 m/s</td>
<td>70</td>
</tr>
<tr>
<td>Thursday</td>
<td>16-03-24(23)</td>
<td>Test</td>
<td>-7 °C</td>
<td>1 mm.</td>
<td>3 m/s</td>
<td>165</td>
</tr>
<tr>
<td>Friday</td>
<td>16-03-25(24)</td>
<td>Training</td>
<td>0 °C</td>
<td>1 mm.</td>
<td>4 m/s</td>
<td>85</td>
</tr>
<tr>
<td>Saturday</td>
<td>16-03-26(25)</td>
<td>Test</td>
<td>0 °C</td>
<td>0 mm.</td>
<td>5 m/s</td>
<td>170</td>
</tr>
<tr>
<td>Tuesday (extra, rain)</td>
<td>16_04_26(25)</td>
<td>Test</td>
<td>0 °C</td>
<td>8 mm.</td>
<td>6 m/s</td>
<td>59</td>
</tr>
</tbody>
</table>

Table 3.8: Dataset division for week 12 (March 21-25) including an extra day which contains rain. The weather data is gathered from [dmi.dk]. Persons denote the number of annotated persons in each video. *Lowest temperature.

As the focus is on persons in different scenarios, each video is split into sequences containing: persons, groups and persons containing reflections. The number of sequences in each video is shown in Table 3.9 and used in the training data.

<table>
<thead>
<tr>
<th>Day</th>
<th>YY-MM-DD</th>
<th>Type</th>
<th>Single</th>
<th>Groups</th>
<th>Reflective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday</td>
<td>16-03-21(20)</td>
<td>Training</td>
<td>28</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>Wednesday</td>
<td>16-03-23(22)</td>
<td>Training</td>
<td>17</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>Friday</td>
<td>16-03-25(24)</td>
<td>Training</td>
<td>-</td>
<td>21</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 3.9: Division into sequences containing single persons, groups or single persons in reflective weather (rain).

Dataset for fall detector / fall predictor and Acceptance Testing

To develop the fall detector and fall predictor, samples of persons falling into the water are needed. Therefore p0-p9 in Table 3.10 is samples of persons falling. The samples marked as training have been used for development.

For testing the overall system this includes the entire pipeline, thermal video accruing, person
3.5. Dataset

extraction, tracking, predict falls and fall detector, a dataset consisting of 99 hours of weekends in April have been used. These videos are recorded in time interval 23 – 07 Thursday, Friday and Saturday.

<table>
<thead>
<tr>
<th>Day</th>
<th>YY-MM-DD</th>
<th>Type</th>
<th>Distance to camera</th>
</tr>
</thead>
<tbody>
<tr>
<td>p0</td>
<td>16-05-19</td>
<td>Training</td>
<td>81.31</td>
</tr>
<tr>
<td>p1</td>
<td>16-05-19</td>
<td>Test</td>
<td>83.82</td>
</tr>
<tr>
<td>p2</td>
<td>16-05-19</td>
<td>Training</td>
<td>43.47</td>
</tr>
<tr>
<td>p3</td>
<td>16-05-19</td>
<td>Test</td>
<td>41.38</td>
</tr>
<tr>
<td>p4</td>
<td>16-05-19</td>
<td>Training</td>
<td>41.26</td>
</tr>
<tr>
<td>p5</td>
<td>16-05-19</td>
<td>Test</td>
<td>42.96</td>
</tr>
<tr>
<td>p6</td>
<td>16-03-03</td>
<td>Training</td>
<td>67.08</td>
</tr>
<tr>
<td>p7</td>
<td>16-03-03</td>
<td>Test</td>
<td>67.93</td>
</tr>
<tr>
<td>p8</td>
<td>16-03-03</td>
<td>Training</td>
<td>67.92</td>
</tr>
<tr>
<td>p9</td>
<td>16-03-03</td>
<td>Test</td>
<td>67.65</td>
</tr>
<tr>
<td>All</td>
<td>April weekends</td>
<td>Test</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3.10: Dataset for developing the fall detector (training) and the acceptance test.

---

3 Video recorded directly from cameras w/o error handling, therefore 14 hours in total are missing (total 120 hours for the 15 nights)

47 hours are removed, since these are used as training/analysis of persons walking at the bridge
4. Person Extraction

The purpose of this module is to extract objects from the thermal video sequences through segmentation. Each of the extracted BLOBs is analyzed, such that a person can be recognized and represented in world coordinates.

The module must fulfill the requirements set in Section 3.3:

1. Must be able to detect and recognize individual persons
2. Must be able to represent a person’s position on the harbor’s pavement plane in world coordinates
3. Must be able to estimate a person’s position when partial occlusion is present
4. Must segment persons from the background
5. Object representation and position should be invariant to reflections caused by rain on the pavement
6. Must be able to extract a person’s position with an accuracy of 5.66 m. in area 1, 2.00 m. in area 2 and 1.00 m. in area 3

4.1 Segmentation

The first step in the pipeline, is the act of detecting and segmenting persons in the image from the background. As described in Section 2.1, the thermal camera forms images based on thermal emittance from objects in the range 190–1000 K (-83 – 726 °C). In order to be able to identify persons in the image, their heat signatures must be different from the background.
Most of the background in the image consists of asphalt, which have an emissivity value close to that of clothes and human skin. Based on the image’s intensity values, the persons are distinguishable from the background as seen in Figure 4.1.

![Figure 4.1: Image of person in video cam_1_16-03-20_23_00_00 (left) and cam_1_16-03-21_06_00_00 (right). The persons are easily distinguishable from the background.](image)

Two general approaches are used for detecting persons in thermal cameras [Jeon et al. 16]:

- A **pixel-based** approach, where pixels are compared frame-by-frame. Examples include modeling each pixel to a Gaussian distribution.

  The advantage of pixel-based approaches is, that it is able to adapt to changes in the image over time (through averaging) and that methods are computationally efficient. Disadvantages include that persons may become part of the background if idle over longer periods of time.

- A **feature-based** approach, where found features are matches against trained templates. These approaches require selection of features which are distinct when comparing regions from the background with those containing persons. Examples include HOG features trained on patches of pedestrian using an AdaBoost classifier [Chang et al. 5].

  An advantage of the feature-based approaches is, that no background model needs to be kept. The disadvantage is, that pre-defined templates of persons must be found through training. The appearance of these templates are affected by changes in weather, such as rain or snow [Jeon et al. 16]. This is partially verified by visual walkthrough of images in rain and regular weather. Two examples are shown in Figure 4.2. Here, the scene is changed such that the image appear blurred and with less contrast during rain. Additionally, due to the changing weather in Denmark, the clothes’ appearances change based on time of the year.

---

1No temperature recordings of the pavement are done on the dataset. Random samples taken at 23:00 in 13 degree weather of pavement where no direct sun has impacted shows readings of 10.43 °C and clothes temperature of 14.4 °C (persons walking for ~30 minutes).
4.1. Segmentation

Figure 4.2: Image of persons in rainy and regular weather from cam_1_16-04-25_23_00_00 and cam_1_16-03-20_23_00_00 respectively.

Due to the computational efficiency and simplicity it has been chosen to select a pixel-based approach. Three different pixel-based approaches are initially considered:

1. **Frame Differencing**: Frame differencing is a technique, which is based on motion caused between consecutive frames. Frame differencing computes the absolute difference image \( I_{\text{diff}} \) between two frames \( I_{f-1} \) and \( I_f \) as shown in Equation 4.1.

\[
I_{\text{diff}} = |I_{f-1} - I_f|	ag{4.1}
\]

The resulting image \( I_{\text{diff}} \) is thresholded to find a binary mask describing the motion. An example of the method applied on two consecutive frames are shown in Figure 4.3.

Figure 4.3: Frame differencing (right) performed on an inputted image (left).

Frame differencing finds small motions between two or more consecutive frames and are highly affected by the object’s velocity [Vahora et al. [38]] in the image plane. Since the camera surveys a large distance of 125 m. objects have different velocities, and thus frame differencing may not be a suitable.

2. **Histogram Based**: The histogram-based technique models the histogram of all pixels. An example of Figure 4.2 is shown in Figure 4.4. Larger values denote warmer temperatures.
As intensities in each pixel describes and object’s temperature, thresholding the image at a given a set interval forms a binary image with only temperatures above the chosen threshold. Ideally, the histogram would have a bimodal distribution containing both foreground and background. The effect of thresholding at five values: 150, 180, 200, 220 and 240 is seen in Figure 4.5.

The person contains values in the range 150-255. From a visual perspective, the best threshold is set at 180-200. In these ranges, buildings and windows emit heat, which segments part of the background as foreground.

3. **Background Model**: The background modeling approach models each pixel to a distribution $\mathcal{G}(\mu, \sigma^2)$ over time. The concept is illustrated in Figure 4.6.

For each pixel, the foreground is found by applying a threshold within a certain distance based on a multiple of the standard deviation $\sigma$. The advantage of modeling each pixel to a Gaussian is that white noise in each pixel can be compensated for. In order to coat with slow illumination changes, the background can be adaptively updated, using
4.1. Segmentation

a simple blending filter as in [Wren et al.][42]. Here, $\alpha$ controls the update speed for an incoming value $y$, such that the mean is updated continuously using Equation 4.2.

$$\mu_t = \alpha y + (1 - \alpha)\mu_{t-1}$$  \hspace{1cm} (4.2)

Adapting the background continuously enable the algorithm to adjust to temperature changes in the scene.

Based on the above description, a background model approach is chosen.

4.1.1 Design and Implementation

The background subtraction is implemented using OpenCV’s `BackgroundSubtractorMOG()` method. The method is based on modeling each pixel to a Gaussian Mixture Model (GMM) over time as proposed in [KaewTraKulPong, Bowden][17]. Here, pixel intensities are modeled as belonging to a certain component of $K$ Gaussian distributions. In three channel images $K = 5$, but can be set to $K = 3$ due to a more reliable thermal image [Szwoch, Szczodrak][35]. Compared to a single Gaussian approach as in Figure 4.6, clustering may more accurately describe patterns in the background by modeling different distributions. Examples include [Friedman, Russell][8] where the two darkest components (in $K = 3$) are background and shadows and the remaining component show a large variance containing foreground.

Pixels are described by weight $w$ belonging to a Gaussian component $G(\mu, \sigma^2)$. The probability of observing the pixel value $x$ is given by Equation 4.3.

$$P(x) = \sum_{i=1}^{K} w_i G(x, \mu_i, \sigma_i^2)$$  \hspace{1cm} (4.3)

For each frame, the pixels are individually compared to the known components ordered by a descending fitness value $w_i/\sigma_i$ [KaewTraKulPong, Bowden][17]. If the distance is within 2.5 standard deviations of the mean, the pixel is matched as belonging to the respective component. The assigned components’ mean $\mu$ is then updated based on $\alpha$ blending. If no matches are found, the distribution with the lowest fitness value is replaced by a new component with high variance and mean corresponding to $x$.

![Figure 4.7: Flow of loading and segmenting each frame in a video.](image-url)
A Video class is implemented, which handles loading of video and processing of frames. When a video is loaded, the first 360 frames are used for learning the background at a blending rate of $\alpha = 0.001$. The flow of the program is shown in Figure 4.7.

### 4.1.2 Evaluation

From visual inspection of the images, the persons are extracted from the background. Requirement 8) **Must segment persons from the background** is fulfilled. Examples of persons are shown in Figure 4.8.

![Figure 4.8: Example of segmented mask (right) from a video (left) using the implemented background subtraction.](image-url)
4.2 Homography Mapping

The theory in this section is based on [Bradski, Kaehler] unless otherwise specified.

As described in Section 2.5, the relation between placement on the camera sensor (in pixels) and coordinates in real-world (in meters) must be found. This projective mapping from image plane to world plane is known as planar homography.

Planar homography has its point of origin in the pinhole camera model. In the pinhole camera model, rays of thermal radiation emitted from an object enter the camera’s pinhole aperture. The image is formed by the rays intersecting the image plane. An illustration of a ray entering the pinhole camera model can be seen in Figure 4.9.

A basic projection of the point \( \mathbf{Q} = [X, Y, Z] \) into the image plane at point \( \mathbf{q} = [x, y, f] \) can be found through the use of similar triangles (e.g. \( \frac{x}{f} = \frac{X}{Z} \)). Ideally, the pinhole and the center of the image plane is placed along the optical axis. In practice, the camera’s pinhole may contain a small offset in both x and y direction on an imager consisting of pixels. Resultingly, \( \mathbf{q} \) physical position in mm. can be found from \( \mathbf{Q} \) on the imager through Equation 4.4:

\[
\begin{align*}
    x_{\text{imager}} &= f_x \left( \frac{X}{Z} \right) + c_x \\
    y_{\text{imager}} &= f_y \left( \frac{Y}{Z} \right) + c_y
\end{align*}
\]  

(4.4)

Where

\( f_x \) and \( f_y \): is defined as the product between the physical focal length \( F \) (in mm) and the pixel size \( s \) (in mm/pixel), such that e.g. \( f_x = F_s \).

\( c_x \) and \( c_y \): is the offset from the optical axis in both x and y direction.

The mapping between \( \mathbf{q} \) and \( \mathbf{Q} \) using Equation 4.4 can be represented using homogeneous coordinates in a single matrix as shown in Equation 4.5:

\[
\mathbf{q} = \mathbf{M} \mathbf{Q} \\
\mathbf{M} = \begin{bmatrix}
    f_x & 0 & c_x \\
    0 & f_y & c_y \\
    0 & 0 & 1
\end{bmatrix}
\]  

(4.5)
Using $\mathbf{M}$ assumes that the points to be mapped are in the same coordinate system as the camera. In practice, persons should not be located in the image plane but in world coordinates. Consequently, an additional mapping between coordinate systems (i.e. “planes”) must be made. An illustration of the concept is shown in Figure 4.10.

besides the perspective transformation, points on an image plane $p_{src}$ can be mapped into the world plane $p_{dst}$ and reversely from world to image plane using a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{r}$. In homogeneous coordinates, this is done using a homogeneous matrix $\mathbf{W} = [\mathbf{R}^t]$. The mapping between $p_{src}$ and $p_{dst}$ is therefore done using the Homography matrix $\mathbf{H}$, where $\mathbf{H} = \mathbf{MW}$. The mapping is shown in Equation 4.6.

$$
\begin{align*}
    p_{dst} &= \mathbf{Hp}_{src} \\
    p_{src} &= \mathbf{H}^{-1}p_{dst} \\
    p_{dst} &= \begin{bmatrix} x_{dst} \\ y_{dst} \\ 1 \end{bmatrix} \\
    p_{src} &= \begin{bmatrix} x_{src} \\ y_{src} \\ 1 \end{bmatrix}
\end{align*}
$$

Where

- $x_{src}$ and $y_{src}$: is the positions on the image plane
- $x_{dst}$ and $y_{dst}$: is the positions on the world plane

4.2.1 Design

The area of interest is Aalborg’s harbor front. Instead of using latitude/longitude coordinates, it is chosen to define a reference coordinate system with origo at camera 2’s placement\(^2\) and $x_1$ direction perpendicular to the harbor edge. Camera 1’s positions is therefore at [16,367.41]. The is shown in Figure 4.11.

To find the Homography matrix $\mathbf{H}$ a set of reference points on the world coordinate plane are

\(^2\)Camera 2 was available for development before camera 1 and thus the origo was placed here.
identified in the camera’s FOV. The world coordinate plane is assumed to be at the same level as the asphalt. Consequently, points can not be mapped reliably, if these are not level with the chosen plane. To locate the points through the thermal camera, the heat signature from a person is utilized. For camera 1, five light poles are used as $x_2$ reference points. Perpendicular to these, up to five points are placed with a 2 m. spacing along the $x_1$ direction. Images containing all reference points and the world coordinate plane are shown in Figure 4.12. From these reference positions, the person’s feet positions are used as reference coordinates ($z = 0$) as listed in Table 4.1.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$p_{1i}$</th>
<th>$p_{2i}$</th>
<th>$p_{3i}$</th>
<th>$p_{4i}$</th>
<th>$p_{5i}$</th>
<th>$p_{6i}$</th>
<th>$p_{1i}$</th>
<th>$p_{2i}$</th>
<th>$p_{3i}$</th>
<th>$p_{4i}$</th>
<th>$p_{5i}$</th>
<th>$p_{6i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>315.73, 0</td>
<td>315.73, 2</td>
<td>315.73, 4</td>
<td>315.73, 6</td>
<td>315.73, 8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>296.60, 0</td>
<td>296.60, 2</td>
<td>296.60, 4</td>
<td>296.60, 6</td>
<td>296.60, 8</td>
<td>296.70, 10</td>
<td>254, 106</td>
<td>229, 107</td>
<td>202, 108</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>276.79, 0</td>
<td>276.79, 2</td>
<td>276.79, 4</td>
<td>276.79, 6</td>
<td>276.79, 8</td>
<td>276.79, 10</td>
<td>208, 89</td>
<td>187, 90</td>
<td>169, 92</td>
<td>145, 94</td>
<td>123, 94</td>
<td>101, 95</td>
</tr>
</tbody>
</table>

Table 4.1: Reference points for camera 1. The world coordinates are denoted in meters with origo at camera 2’s position. Positions marked with "-" are either occluded or not reachable and not included as reference points.

The chosen reference points and plane is shown in Figure 4.12.

The homography for camera 1 is created using the points in Table 4.1. The trained model is tested by re-evaluated by applying the Homography matrix to the $p_{dst}$ positions.

### 4.2.2 Implementation and Evaluation

The Homography matrix is calculated using OpenCV’s `getHomography`-function, which performs a least-squares regression based on inputted points and outputs the $3 \times 3$ Homography matrix $H$. The Homography matrix is shown in Equation 4.7.
Figure 4.12: Reference positions as viewed in the thermal camera for camera 1. The positions of the feet are used as reference positions.

\[
H_{cam1} = \begin{bmatrix}
0.121 & -0.378 & 8.462 \\
-0.814 & -8.667 & 495.191 \\
-0.003 & -0.023 & 1
\end{bmatrix} \tag{4.7}
\]

The difference between the calculated \( p_{src} \) and known \( p_{src} \) is listed in Table 4.2. The mean error for camera 1 is 0.30 m. The reason why the positions deviate, is due to pixel round off errors when selected the person’s feet positions. These effects should though be negated through use of least-squares regression. Requirement [3]. Must be able to represent a person’s position on the harbor’s pavement plane in world coordinates is fulfilled.

<table>
<thead>
<tr>
<th>( i )</th>
<th>( p_{i,1} )</th>
<th>( p_{i,2} )</th>
<th>( p_{i,3} )</th>
<th>( p_{i,4} )</th>
<th>( p_{i,5} )</th>
<th>( p_{i,6} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>0.23</td>
<td>0.16</td>
<td>-</td>
<td>0.39</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>0.36</td>
<td>0.14</td>
<td>0.16</td>
<td>-</td>
<td>-</td>
<td>0.54</td>
</tr>
<tr>
<td>3</td>
<td>0.04</td>
<td>0.11</td>
<td>0.22</td>
<td>0.61</td>
<td>0.54</td>
<td>0.37</td>
</tr>
</tbody>
</table>

Table 4.2: Distance difference between calculated world position and known world position in meters for camera 1. The world positions are calculated from the homography mappings in Equation 4.7.

The homography mapping for camera 2 is shown in Appendix B.
4.3 Position extraction

Based on the binary image obtained from the background subtraction part described in Section 4.1 it is of interest to extract a person’s position as a pixel coordinate. The position will be mapped using the planar homography mapping described in Section 4.2 to world coordinates. As already described the mapping is most accurate at the pavement’s plane, requiring extraction of the feet’s position on the ground. In order to find a person’s feet position several steps need to be taken:

1. **Moving object detection**: first step will be to detect persons in the scene. The background subtraction provides a binary mask of the scene where the moving parts are marked with white pixels (foreground). Some of these foreground pixels will be temporal noise or moving objects. The moving objects will most likely be persons (running, walking or biking as described in subsection 2.2). Moving objects objects of interest should therefore be larger regions of connected foreground pixels. In order to find these connected areas the algorithm for finding larger binary objects in an image will be used (BLOB detection – Binary Larger OBjects).

2. **Person separation logic**: as described in subsection 2.2 different “events” occur in the scene. This can be persons walking close together, resulting in an area where foreground pixels from multiple persons are connected (BLOBs are too wide). Besides BLOBs being too wide a BLOB can also be too tall in cases where a person partial occludes another person or if reflections from puddles are present. This can cause the feet position to slide and cause inaccuracies. Different approaches exist to separate multiple persons in the same BLOB. In Jeon et al. [16] a histogram is created for each row, indicating the number of foreground pixels present. Here, a split is performed where the smallest number of pixels is found. Another similar approach is done in Gade et al. [9] where the BLOB’s convex hull is analyzed and splitting is done at found convexity defects. Gade et al. [9] is used as basis for the person separation, with an extended analysis of parameters.

The following sections will be structured as follows:

- **Single person detection** including initial logic to remove noise.
- **Wide BLOB splitting** to separate a BLOB if multiple persons are connected horizontally.
- **Tall BLOB splitting (a)** to separate a BLOB if a person and its pavement reflection is connected vertically.
- **Tall BLOB splitting (b)** to separate a BLOB if multiple persons are connected vertically.

The 4 scenarios are shown in Figure 4.13.
Chapter 4. Person Extraction

Figure 4.13: The 4 scenarios which the person extraction algorithm will be able to handle are single person (left), wide splitting of multiple persons who are connected horizontally (middle-left), tall splitting of person and reflection which are connected vertically (middle-right) and tall splitting of multiple person who are connected vertically.

The desired position representation is shown in Figure 4.14 by the green dot near the feet’s position. This position can be extracted by finding the bounding box around the person. Ideally, the center position of the bottom line represents the person’s feet position.

Figure 4.14: The algorithm should be able to find the feet position of persons (green dot). As seen this position is approximately at the center of the person (shown by the red line) at the lowest part of the person.

4.3.1 Single person

This section will handle all single persons. When the binary mask is obtained from the background subtraction, it is assumed that the biggest BLOBs have to be persons in the scene. Despite the background subtraction, BLOBs containing non-persons still occur – therefore smaller BLOBs have to be removed, assumed to be noise. The remaining part of this section will describe:

- Finding BLOBs of foreground objects
- Non-person BLOB removal, which removes BLOBs to small to be persons
- Recognize persons and determine feet position
- Height ratio estimation to remove BLOBs containing small objects such as birds and dogs.
- Static occlusion handling
4.3. Position extraction

Finding BLOBs of foreground objects

First step is to identify all BLOBs in the binary image. To find all BLOBs in an image, connected component analysis is used [Szeliski, 2010]. Connected component analysis works by going through the entire image finding all regions where neighboring pixels have the same value. In Figure 4.15 (left) an example of the binary mask of a single person walking in the scene is shown. Using connected component analysis (8-connectivity) will result in 12 different regions, shown by the red arrows in Figure 4.15 (right).

![Figure 4.15: A background segmented image (left) results in 12 BLOBs found denoted with arrows (right).](image)

Non-person BLOB removal

In the binary mask, some of the regions found by a connected component analysis might be noise. In example Figure 4.15, 11 of the BLOBs are smaller regions and should be considered as noise – leaving only the person. This requires setting a static threshold for a minimum BLOB size of a person.

According to Johnsons Criteria described in Section 3.1 it should be possible to recognize persons at a distance of 125 meters from the camera. As described in Subsection 3.1.5 this corresponds to row 76 in the image. Resultingly, the system should not try to find positions of persons in the image walking in the area above row 76. Figure 4.16 shows the maximum distance where the system can operate.

![Figure 4.16: Red line showing the maximum range of 125 meters from the camera, corresponding to row 76. This forms the basis for a minimum BLOB size area for a person. A blue BLOB of 20 pixels is shown in the mask (right) denotes the minimum BLOB area for a person.](image)
Using the maximum operating distance, it is possible to determine a minimum pixel area of a person. The BLOB containing the person shown in Figure 4.16 is 58 pixels in area and has an estimated height of 2 meters. People vary in height and width, so in order to remove minor areas in the scene by a rather simple step, it has been decided to remove BLOBs, which have an area smaller than 20 pixels. To give the reader an idea of an area of 20 pixels, a blue BLOB has been added to Figure 4.16 shown next to the person.

**Feet position**

After having removed smaller regions known to be noise, a bounding box can now be found for each of the remaining person candidates (BLOBs) in the image.

As described, it is desired to estimate the feet position of a person near the pavement plane. The position can be found by from the bounding box around the BLOBs and is assumed to be at the center of the bottom side of the rectangle (lowest side with highest y-value in the image). In Figure 4.17 the bounding box and feet position for the person is shown.

![Figure 4.17: Bounding box around the person (red) and extracted feet position.](image)

**Height estimation**

Creating bounding boxes around BLOBs over 20 pixels in area does not only find persons, but also animals such as dogs and birds. Since dogs, birds and other animals are not of interest some logic need to handle such cases. Due to the rather big difference in height between a person and animals this information is used to make the logic discard lower candidates.

The height of BLOBs can be estimated using the found feet position, homography and pinhole camera model. As described in Section 4.2 the pinhole’s similar triangles describe the relation between the object in the real world and on the image plane. By knowing the distance from the camera to the person $F$, focal length of camera $f$ and physical pixel size on the image sensor $s$, the BLOB’s height can be calculated, as shown in Equation 4.8 and derived from Section 4.2.

$$X = \frac{F \cdot x}{f} \cdot s \quad (4.8)$$

Where
4.3. Position extraction

\( x \): is the person height in pixels on the camera sensor

\( f \): is the focal length of the camera in mm.

\( X \): is the person’s height in m.

\( F \): is the distance from the camera to the person in m.

An example of a dog in the scene is shown in Figure 4.18. By estimating the height of the dog it is possible to set a threshold removing all candidates which are too small. In the given example the dog is 0.79 meters tall compared to the person who is 1.70 meters tall.

Figure 4.18: Example of positions found from bounding boxes around BLOBs. This also finds dogs, which is unwanted.

To find a threshold, the average BLOB height in meters for individual persons in training set are found (not using videos from 16_03_25 containing reflections from puddles) and the distribution is shown in Figure 4.19.

![Average person height](image)

Figure 4.19: BLOB height for persons in the dataset (omitting data from reflection videos) represented in meters. The dotted lines show the 99.7% confidence interval.

Heights are measured to be in the range \(~1.41–2.27\). It needs to be noted that the height results can be biased by:

- Pixel roundoff
• Errors in segmentation
• Errors in homography

By using a the lower confidence bound of 99.7% as a threshold (1.41 meter) lower BLOBs such as dogs are removed.

Static occlusion handling

Different static object in the scene might cause inaccuracies in the extraction of the feet position. Since the feet position is based on the bounding box around the person, horizontal occlusion Figure 4.20 (left) and vertical occlusion Figure 4.20 (right) might either remove the feet or push the position to the sides.

Figure 4.20: Left image shows a person being horizontally occluded by the pole placed in the scene. This causes the feet position to move to the left. Right image shows a person being horizontally occluded by a sign, which causes the feet position to move “up in the image” meaning the person seems to be further away from the camera.

To avoid getting inaccurate positions caused by an inconsistent BLOB (partially occluded) the width/height ratio for a BLOBs is used. BLOBs smaller than a set threshold should thus be removed. Training data, containing single persons are used (not using videos from 16_03_25 containing reflections from puddles). For the test not to be biased by different areas, the frames have been divided into 3 different types:

• Normal frames: the entire person.
• Occlusion frames: person partially or totally occluded.
• Entry/exit frames: person in an entry or exit area are removed.

In Figure 4.21 occlusion areas and entry/exit areas are marked.
4.3. Position extraction

In Figure 4.21 two histograms of width/height ratios are shown. For both histograms the 99.7% confidence interval is found and illustrated with dotted lines. Setting a high threshold will allow more square persons in the scene, whereas a low threshold allows highly rectangular persons. Setting a high threshold will cause more inaccurate positions, whereas a low threshold will remove more positions from the algorithm. Therefore the lower bound from the confidence interval 20.35 from normal frames is used to determine when the ratio between the width and height is unlikely to be an entire person.

![Histograms showing width/height ratios for normal and occlusion frames.](Image)

**Figure 4.22**: Two histograms showing the distribution of how the ratios change during the training data of single persons. Based on the mean and standard deviation of the data, a 99.7% confidence interval is found for each histogram. The upper and lower bound are visualized by the dotted lines.

**Ignore person in entry/exit areas**

Inaccuracies in the position might occur in case partial occlusion by static objects occur. Since the reason is that the BLOBs are not consistent, inaccuracies might also occur in case a person is entering or leaving the scene. Therefore, all BLOBs which are in such an entry/exit area should be ignored. To do this, the same mask as shown in Figure 4.21 (right) used in the test to find the width/height ratio should be used to discard positions from persons which are in these areas.
Evaluation

The algorithm is now able to handle single persons in the scene, this includes removing non-person BLOBs, noise and persons in occlusion areas.

To develop this module, the training data has been used to see how accurate the positions are. The training data contains single persons, where very little reflections from pavement should occur. In Figure 4.23, the mean pixel error (blue bars) and variance pixel error are shown. These are plotted for each of the training sequences. The pixel error is the Euclidean distance between the calculated and annotated position.

Pixel Error – Single Persons (Training data)

![Pixel Error Chart]

Figure 4.23: Showing pixel error between calculated position (by algorithm) and manual annotated position. The left bar plot (blue) shows the mean and the right bar plot (red) shows variance pixel error for each sequence. The overall mean of the pixel error is: 2.97

During the test it has been observed that the interpolation between sample points in the
annotated positions might slide a bit during the sequence. This can lead to a higher error than if the positions had been annotated manually for each frame. The overall mean of the pixel error is: 2.97 for all the sequences.

4.3.2 Wide-splitting of persons

In previous section the steps to handle single persons were described and evaluated. Since multiple persons appear in the scene at the same time, the persons may occlude each other and form one large BLOB. In Figure 4.24 an example is shown where two persons walk close to each other and forms one BLOB. As illustrated with the green dot, only one position is found.

Figure 4.24: Shows how two persons are connected into one big BLOB (left image). From the single person handling this seems to be only one person, shown in the right image by the green dot.

There are different ways a split can be performed:

- Based on features
- Based on convexity information

Feature based could for example be by trying to find circular objects, since these would often be the head of a person or try to identify body parts, such as legs. As seen in Figure 4.25 the BLOB may not contain any texture which makes identifying parts difficult.

Figure 4.25: Example of a BLOB containing two persons occluded horizontally.

Another way to perform the split, is to analyze the convexity defect between the heads [Gade et al. 9]. The heads will often standout from the BLOBs due to the fact that the head is smaller than the torso. Therefore, when persons horizontally occlude each other there will be
a favorable splitpoint approximately until almost totally occluded. In Figure 4.26 two persons occluding each other at different distances are shown.

![Figure 4.26](image)

**Figure 4.26:** Two persons connected where the blue person partially occludes the white person more and more (from left to right). As seen the heads are still able to be separated.

Another advantage of using this method is that the same can be used to split tall BLOBs along the vertical dimension.

The approach firstly finds the convex hull of the contour and then separates the two persons by the highest convexity defect in the upper part of the BLOB. A convex hull is the minimum convex polygon covering an entire BLOB [Moeslund]. After having the convex hull of the contour convexity defects should be analyzed. Knowing that the heads are in the upper part of the BLOB this will result in a large convexity defect. A convexity defect is an area in the convex hull where the interior does not belong to the contour. The red marked areas Figure 4.27 (middle) are defects of the convex hull. The point where the convexity defect with the farthest distance to the boundary (from the boundary which has the lowest y-value) will be the best candidate to separate the two persons. This point is shown by the red arrow in Figure 4.27 (right).

![Figure 4.27](image)

**Figure 4.27:** Example of a BLOB’s convex hull (left), its convexity defect area (middle, red) and the split point at the highest defect (right).

**Implementation**

The implementation description of the wide splitting algorithm is separated into the following sub parts.

1. **Determining if a wide split should be performed**
2. **Finding an optimal split point**
4.3. Position extraction

Determining if a wide split should be performed When having extracted the feet position from the previous module single person a threshold should be used to indicate if a contour is too wide to be only one person. By using the width/height ratio test results from Figure 4.22, the upper bound (55.44\%) is used as a threshold.

In addition to the width/height check, the contour perimeter should also be larger than the bounding box perimeter as described in \[Gade et al.\] 9. To support the idea of using the perimeter as a constraint for making a wide-split Table 4.3 shows the convex hull and bounding box perimeter for two single BLOBs C1 and C2 and as a combined BLOB C1+C2.

<table>
<thead>
<tr>
<th>BLOB</th>
<th>Convex hull perimeter</th>
<th>Bounding box perimeter</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>215</td>
<td>230</td>
</tr>
<tr>
<td>C2</td>
<td>253</td>
<td>224</td>
</tr>
<tr>
<td>C1+C2</td>
<td>467</td>
<td>296</td>
</tr>
</tbody>
</table>

Table 4.3: Shows the convex hull perimeter vs the bounding box perimeter. As in \[Gade et al.\] 9 the convex perimeters indicates that a split is need, since when two persons are connected the convex hull perimeter will typically be significantly higher.

Equation 4.9 illustrates the constraint.

\[
(55.4 \% < BB_{aspectRatio}) \quad \& \quad (O_{perimeter} > BB_{perimeter})
\]  (4.9)

Where

- \(BB_{aspectRatio}\): is the BLOB’s bounding box aspect ratio, \(\frac{BB_{width}}{BB_{height}}\).
- \(BB_{perimeter}\): is the bounding box’s perimeter in pixels, \(2 \cdot BB_{width} + 2 \cdot BB_{height}\).
- \(O_{perimeter}\): is the object’s perimeter in pixels.

Where width and height of the contour is found using OpenCV’s \texttt{boundingRect} on the current BLOB. This function returns the x and y coordinate for the upper left corner of the bounding box, together with the width and height in pixels of the BLOB. To find the perimeter \texttt{arcLength} is also applied to the current BLOB, which returns the perimeter length in pixels.

Finding the optimal split point In case the criteria in Equation 4.9 is fulfilled, the BLOB needs to be split. To do this the convexity defects of the convex hull is used. To find the convex hull of the BLOB OpenCV’s \texttt{convexHull} is used. This function returns the indexes of the BLOB, which describes the points used to construct the convex hull. Passing these indexes to \texttt{convexityDefects} the following information about each defect is obtained:

- Start and end point of BLOB line representing a part of the convex hull
- Largest defect point
- Distance from the line to the largest defect point

Since the idea is to split by the largest convexity defect found from the upper part, the list containing all the lines are sorted, in order to find the line segment with the lowest y-value (can be either the start or end point of a line). An additional constraint is added so that in
order to be a valid split point candidate, the line segment has to have a gradient below 1.5. This will help discard candidates like the red shown in Figure 4.28.

**Figure 4.28:** Illustration of finding the largest convexity defect from the line segment with the lowest y-value.

This constraint means that the line segment is not allowed to have a slope of more than $56.3^\circ$. The gradient is calculated by Equation 4.10:

$$\text{gradient} = \left| \frac{(end_y - start_y)}{(start_x - start_x)} \right| \quad (4.10)$$

Where

- *start*: is the line’s starting point.
- *end*: is the line’s end point.

After having found all possible candidates which fulfills the above, constraints are now sorted in order to find the one with the defect point farthest away from the line.

When having an optimal split point for the current BLOB, the feet position of the left person is found by using the $BB_{x,y,height}$ and $SplitPoint_{x,y}$, using the following Equation 4.12:

$$feetPosition_{left, person_x} = BB_x + \frac{SplitPoint_x - BB_x}{2} \quad (4.11)$$

$$feetPosition_{left, person_y} = BB_y + BB_{height} \quad (4.12)$$

By using the previous equation results in the feet position for the left person as shown in Figure 4.29. As seen in Figure 4.29 (left) additional unwanted zero rows between the person and its feet position (the green dot) exists. To remove this spacing, boolean indexing in Python is used on the patch around which encapsulates the person.
4.3. Position extraction

**Figure 4.29:** Left figure shows the two persons, where the feet position for the left persons is found – but has additional zero rows, since the right person has its feet lower in the image. Right figure shows only the left person, where the left part is where the additional rows remain and in the right part these zero rows have been removed.

Since more than two person can be connected in the same BLOB, only the position of the left person is stored as a position. The found person is removed from the mask. In **Figure 4.30** the found person before and after removal is shown.

**Figure 4.30:** Left figure shows the current contour of interest where the feet position of the left person has been found. By removing this found person from the area – shown in the right image, the new ROI can be researched for contours.

The remaining mask can now be researched for new BLOBs. All newly found BLOBs are propagated back to the beginning of the algorithm for processing again. By doing this, multiple persons can now be handled. In the next paragraph the training data will be used to see how accurate the positions are.
Figure 4.31: Showing the flow of the die splitting: 1) POP a BLOB from the stack, 2) Split the BLOB and store position for the person to the left of the split point, 3) Find BLOBs in the new image, 4) PUSH the new BLOBs to the stack.

The algorithm is able to handle single persons and persons who are connected horizontally. The accuracy of the wide splitting will be evaluated in the next paragraph.

Evaluation

In the training data the sequences where connected BLOBs occur have been identified and the wide splitting algorithm has been tested. In the Figure 4.32 the sequences are shown with their pixel error mean and variance.
4.3. Position extraction

Pixel Error – Wide Split Persons (Training data)

Figure 4.32: Showing pixel error between calculated position (by algorithm) and manual annotated position. The left bar plot (blue) shows the mean and the right bar plot (red) shows variance pixel error for each sequence. The overall mean of the pixel error is: 5.71

The overall pixel error is 5.71. From the Figure 4.32 there are some sequences which have a rather large mean and/or variance. These sequences have been visually analyzed, in order to identify the causes of the increased pixel error. The following items sums up the overall errors:

- Reflections from pavement
- Sliding annotated position due to interpolation errors
- BLOB separation from bikes
- Bad sequence start – caused by a bad background segmentation. If a person is in the scene, the person will be a part of the background a few frames

`cam_1_16_03_21_06` – sequence 6:
In this sequence a single person causes the pixel error to increase due to reflections from the pavement together with a sliding position annotation. These examples are both shown in Figure 4.33.
Figure 4.33: Blue circles shows position from the algorithm, red circles shows annotated positions. Left figure show that reflections from the pavement causes inaccurate position. In the right figure a sliding annotation causes the pixel error to increase.

As seen in Figure 4.34 the wide splitting works.

Figure 4.34: Blue circles shows position from the algorithm, red circles shows annotated positions. The position for both the single person and the two connected persons are found.

cam_1_16_03_24_23 – sequence 11:
In Figure 4.35 (left) it is shown that the wide-splitting algorithm works. In Figure 4.35 (right) the splitting is still working, but the water puddles cause reflections – which results in an increased pixel error.

Figure 4.35: Blue circles shows position from the algorithm, red circles shows annotated positions. Left figure showing wide splitting works, where the two persons to the right in the image is separated. Right figure shows that puddles on the pavement will cause reflections from the persons, which will make the positions in-accurate.

In the same sequence two persons walk close to each other and in Figure 4.36 (left) it is observed that in some frames the bike causes the person to be split into two parts. Further more in Figure 4.36 (right) the annotation is sliding – both of these mentioned events, causes again an increase in the pixel error.
4.3. Position extraction

**Figure 4.36:** Blue circles shows position from the algorithm, red circles shows annotated positions. Left figure shows how a bike can cause a person to be cut into two parts, since the heat from the bike is much different from the persons heat signature. Right figures shows how the annotated positions sometimes might slide a bit, this will also cause the pixel error to seem bigger than in actually is.

cam_1_16_03_25_00 – sequence 3:
As seen in [Figure 4.37](left) the wide-splitting works but the reflections from pavement cause the positions found to be inaccurate. In [Figure 4.37](right) the reflections together with the example being rather complex caused by the 4 persons occluding each other also causes the pixel error to increase.

**Figure 4.37:** Blue circles shows position from the algorithm, red circles shows annotated positions. Left figures shows how a bike causes an unwanted cut of the left person which results in a position not being found, and the right person has no more annotated positions. Right figure shows a sliding annotated position.

In [Figure 4.38](left) the group of persons walks behind a parked car which causes one of the persons not to be found, and since this person is close to the window, a reflection is here found to be a position of a person. In [Figure 4.38](right) one of the persons is still behind the pole (not together with the rest of the group leaving the scene) resulting in a found position from the third person leaving the scene, being compared with the person annotated behind the pole. Hereby a rather high pixel error will be found.

**Figure 4.38:** Blue circles shows position from the algorithm, red circles shows annotated positions.

cam_1_16_03_25_00 – sequence 12:
The wide splitting seems to be very robust, but the reflections from pavement results in high inaccuracy as shown in [Figure 4.39](left)
Chapter 4. Person Extraction

Figure 4.39: Blue circles shows position from the algorithm, red circles shows annotated positions. Reflections causing inaccuracy.

*cam_1_16_03_25_02 – sequence 1:* Pavement reflections and sliding annotations causes the inaccuracy as seen in Figure 4.40. It was also observed that in 4 frames reflections was found in the windows (and the persons occluding each other too much – leading to no split), which caused an inaccuracy around 100 pixel to be found.

Figure 4.40: Blue circles shows position from the algorithm, red circles shows annotated positions. Reflection from pavement and windows causing high pixel error.

*cam_1_16_03_25_03 – sequence 1:* This sequence is where 3 persons walk close to the boundary of the search area. In Figure 4.41 (left) the background model is not correctly initialized, causing a found person in the binary image, but looking at the thermal image, the person is no there. This happens in the two first frames, resulting in an error of around 48 pixels. This

Figure 4.41: Blue circles shows position from the algorithm, red circles shows annotated positions. Left image shows that the background model is not correctly initialized. Right image shows that the splitting finds 4 individual persons.

Test recap: This test shows that the wide splitting algorithm works as expected. The increased pixel error is mainly caused by reflections from the pavement. Furthermore it is observed that reflections in the windows can occur – but only shortly causing the algorithm to find extra unwanted positions.

To cope with the problems caused by the reflections a tall split module will now be designed and implemented.
4.3.3 Tall-splitting of persons

As persons walking close together cause BLOBs to become too wide, a BLOB can also become too tall. This occur if two persons walking at different distances to the camera pass by each other or reflections from the pavement causes the background subtraction to find foreground. A tall BLOB which causes wrong positions to appear is shown in Figure 4.42.

Figure 4.42: Illustration of a wrong position assigned to the person left in the image. Due to the water on the pavement, reflections will appear.

In order to determine if a BLOB is too tall it is necessary to know the height of the person. Obtaining the height at a given position in the scene was described in Subsection 4.3.1. In Subsection 4.3.1 the average heights were obtained using the training data. In the test shown in Figure 4.19 the upper bound (2.27 m) from the confidence interval should be used as the threshold to decide whether a person is too tall. Therefore when a BLOB taller than 2.27 m. the algorithm should try to either:

- Check if the tall BLOB was caused by reflections from the pavement Figure 4.13 (middle/right)
- Split two persons walking at different distances to the camera, causing a BLOB to become too tall Figure 4.13 (right)

In order to perform the splitting of tall BLOBs, the information from the convexity defects will again be used. The largest convexity defect seen from the left or right side will be used to find which of the line segments are of interest. Consequently, the gradient should be larger than 1, since this will cause the slope to be larger than 45° and thus avoid finding a split point between the legs.

Having identified the best splitting point a test should be performed to check if the lower part of the BLOB is a reflection. To do this the approach described in Gade et al. [9] will be used. This method is based on the knowledge that the reflections which will occur, are created by the persons, as described in Section 2.1.

Since the BLOB is separated into two, the lower part should be vertically flipped and matched against the upper region. If the two region’s foreground pixels match, the lower region will be removed and only feet positions for the upper part will remain. In Figure 4.43 a zoomed and cropped version of Figure 2.4 shows how the binary mask will be obtained. As seen the pavement causes a reflection which is connected to the BLOB from the person (left).
Figure 4.43: Left figure shows how the water on the pavement can cause a reflection which is connected to the BLOB from the person. The right figure shows the ROI (green rectangle), the ideal split between the two areas (red dotted line) and the lower part which has been mirrored up into the upper region (blue area).

Ideally the BLOB should be split close to the feet where the convexity defect is the largest. The ideal split, together with the idea of vertically flipping the lower region and matching against the upper, is shown in Figure 4.43 (right)

Implementation

The implementation of the tall-splitting algorithm is split into the following parts:

- Identify line segment candidates
- Find the largest convexity defect, without intersecting the foreground
- Find the two possible feet positions
- Check if the lower feet position is caused by pavement reflection

Identify line segment candidates  To find the line segments which has the largest convexity defect, the same approach as described in Subsection 4.3.2 will be used. The only difference is, that it is desired to find line segments with an absolute gradient above 1. In Figure 4.44 (left) the red, green, blue and purple line segments are therefore candidates to search for largest defect.
4.3. Position extraction

Figure 4.44: Red, green, blue and purple line segments satisfy the constraint of an absolute gradient above 1.

By using the information obtained from OpenCV’s `cv2.convexityDefects` the largest defect is found from the green line segment shown in Figure 4.44 (middle).

Recalling how the wide-split was performed, the largest defect point was used as splitting point. This works fine when separating between the heads, but if it used directly as splitting point for the tall split, the split might cut off some part of the legs/feet of the person. An example where the largest defect is used as split point is shown in Figure 4.44 (right). To avoid this wrong splitting of objects, some more logic needs to be used.

**Find the largest convexity defect, without intersecting the foreground**

To avoid intersecting some part of the person’s legs a search is performed along the line segment and towards the foreground pixels. The correct splitting point is then found from the row with the largest number of zero pixels (searching from line segment until the column of the largest convexity defect). The row with the largest number of zero pixels yield the largest defect, seen from the side. The corrected splitting point is shown by the blue dot in Figure 4.45.

Figure 4.45: Using a search for the row (from line segment to largest defect) of largest number of zero pixels, the corrected tall splitting point is shown by the blue dot. The red dot shows the largest convexity defect from the `cv2.convexityDefects`.

**Find the two possible feet positions**

The two possible feet positions are store by:

\[
\begin{align*}
feetPosition_{upper} &= feetPosition_x, feetPosition_y - \text{diff}(feetPosition_y, splitPoint_y) \quad (4.13) \\
feetPosition_{lower} &= feetPosition_x, feetPosition_y
\end{align*}
\]
Where

$$feetPosition_x, feetPosition_y$$ is the initial feet position found by single person algorithm.

The upper and lower regions are stored as patches, for further investigation.

**Check if the lower feet position is caused by pavement reflection** To perform the reflection check the function reflectionCheck is implemented. The function inputs two patches (patch1 = upper region, patch2 = lower region) found from the tall split algorithm as input. In order to check if the lower patch is a reflection in the pavement, the following steps should be done:

- Vertical flip patch2 and add zero matrix on top to make equal dimensions, check if the overlap is above 80%
- Move the patch2 1 pixel to left, check if the overlap is above 80%
- Move the patch2 1 pixel to right, check if the overlap is above 80%

Initial test showed that using the threshold of 90% as described in [Gade et al. 9], reflections was not detected as properly, therefore an additional 10% was added yielding a threshold of 80%.

To check how many percent of the foreground pixel which match, a bitwise AND is performed between the two patches, as shown in **Equation 4.15**

$$tempPatch = patch1 AND patch2$$  

(4.15)

Afterwards, a division between the sum of foreground pixels in the tempPatch and patch1 is performed as shown in **Equation 4.16**

$$reflection_{percentage} = \frac{\text{sum}(tempPatch_{foreground, pixel})}{\text{sum}(patch1_{foreground, pixel})}$$  

(4.16)

Where

$$reflection_{percentage}$$ yields the number of foreground pixels between the upper and lower patch’s match/overlap.

In case the percentage is not larger than 80% – the steps above are performed again, but where patch2 is shifted to both sides (performed in two iterations). If one of the 3 checks overlaps by more than 80%, the lower feet positions found by the tall split algorithm are discarded. If the regions are not marked as a reflections, both of the feet positions are stored and outputs two persons. An example is shown in **Figure 4.46**.
4.3. Position extraction

Figure 4.46: Example of a BLOB which have been split by the tall splitting algorithm (shown by the red dotted line). The lower part of the BLOB is mirrored in the red dotted line (shown by the blue area). This blue area should are compared to the upper region and if the foreground pixel overlap more than 80 % the lower region is caused by an reflection.

Evaluation

In the training data the sequences where BLOB’s reflections occur have been identified and the tall splitting algorithm has been tested. In the Figure 4.47 the sequences are shown with their pixel error mean and variance.
Chapter 4. Person Extraction

Figure 4.47: Left plot is without tall splitting enabled – overall mean 5.04. Right plot is with tall splitting enabled – overall mean pixel error 4.47

16_03_24_23_seq_13 (High mean, low variance)
Mean: 7.14
The reason for having a mean pixel error of 7.14 is, that the height calculation for the persons is not exceeding the threshold of 2.27 m, therefore although reflections from the pavement causes difference in annotated and extracted position. An example where the height is below the constraint for splitting is shown in Figure 4.48 (left) and an example of where the tall splitting works and handles reflection is shown in Figure 4.48 (right).

Figure 4.48: Blue circles shows position from the algorithm, red circles shows annotated positions.

16_03_25_00_seq_13
Mean: 6.98
The sequence is rather short only around 10 seconds (the person walks out of building and into a parked car). Wrong split points are found as shown in Figure 4.49 (left) where a larger defect is found very close to the boundary of the water puddle. In Figure 4.49 (right) an example where the BLOB is too tall caused by a reflection is handled correctly.
Recap:
The person extraction algorithm has now been implemented and should now be able to handle the following cases:

- Extract the feet position of a single person walking in the scene.
- In case the single person is found to wide – indicating that more than one person are partially occluding each other, the BLOB is split horizontally, and the individual feet positions are found.
- If the BLOB is found to tall – indicating that more than one person are partially occluding each other, the BLOB is vertically. If the lower region is detected as a reflection, this regions is removed and only the upper region (the person) gets a position assigned.

Since handling of multiple persons occluding each other can be very complex, different pixel errors or missing positions will occur. In the next section a module test will test the performance of the person extraction module.

4.4 Module Test

The purpose of the test is to satisfy the following requirements:

2) Must be able to detect and recognize individual persons
4) Must be able to extract a person’s position with an accuracy of 5.66 m. in area 1, 2.00 m. in area 2 and 1.00 m. in area 3
6) Must be able to extract a person’s position when partial occlusion is present
9) Object representation and position should be invariant to reflections caused by rain on the pavement

The module test is performed on the test data, which as described in Section 3.5 contains in all 30 hours of video (461 annotated persons).

The position accuracy is measured by comparing the outputted positions to the ground truth positions, which may result in three scenarios:

- Number of annotated positions > Number of outputted positions: Occurs if the algorithm fails to detect and recognize persons (FN), or if persons are partially or fully occluded.
• **Number of annotated positions = Number of outputted positions**: The algorithm works as expected.

The outputted positions must be associated with their ground truth. Since the dataset contains 2,880,000 frames across 32 hours of video, this is deemed almost impossible to manually associate for each frame. Instead, it is chosen to associate positions using the nearest neighbor.

**Procedure**

1. **Run** `main_extractPositions.py` **to extract the positions from the test data sequences.**

2. **Run** `test_scripts/module_segmentation_evaluate.py` **to find all**
   - **False-Negatives**
   - **Distance between annotated and algorithm position (error)**
   - **Sequence length**

**Results**

In **Table 4.4** the results from the test are shown. The results for each sequence are attached in **Appendix C**. The table shows the number of FN (in %), error distance between annotate - d vs. algorithm positions (error in meter) and lastly the sequence length (in seconds).
4.4. Module Test

<table>
<thead>
<tr>
<th>Sample</th>
<th>FN [%]</th>
<th>Error [meter]</th>
<th>Sequence Length [seconds]</th>
</tr>
</thead>
<tbody>
<tr>
<td>cam_1_16-03-21_23_00_00.mkv</td>
<td>11.83</td>
<td>0.42</td>
<td>511.44</td>
</tr>
<tr>
<td>cam_1_16-03-22_00_00_00.mkv</td>
<td>30.94</td>
<td>0.90</td>
<td>129.56</td>
</tr>
<tr>
<td>cam_1_16-03-22_01_00_00.mkv</td>
<td>16.69</td>
<td>0.34</td>
<td>260.48</td>
</tr>
<tr>
<td>cam_1_16-03-22_02_00_00.mkv</td>
<td>13.57</td>
<td>0.27</td>
<td>65.16</td>
</tr>
<tr>
<td>cam_1_16-03-22_03_00_00.mkv</td>
<td>13.88</td>
<td>0.27</td>
<td>142.64</td>
</tr>
<tr>
<td>cam_1_16-03-22_04_00_00.mkv</td>
<td>20.10</td>
<td>0.24</td>
<td>100.08</td>
</tr>
<tr>
<td>cam_1_16-03-22_05_00_00.mkv</td>
<td>13.80</td>
<td>0.27</td>
<td>124.68</td>
</tr>
<tr>
<td>cam_1_16-03-22_06_00_00.mkv</td>
<td>36.77</td>
<td>0.29</td>
<td>646.64</td>
</tr>
<tr>
<td>cam_1_16-03-23_23_00_00.mkv</td>
<td>10.34</td>
<td>0.45</td>
<td>793.2</td>
</tr>
<tr>
<td>cam_1_16-03-24_01_00_00.mkv</td>
<td>16.02</td>
<td>1.02</td>
<td>578.92</td>
</tr>
<tr>
<td>cam_1_16-03-24_02_00_00.mkv</td>
<td>22.82</td>
<td>0.15</td>
<td>154.24</td>
</tr>
<tr>
<td>cam_1_16-03-24_03_00_00.mkv</td>
<td>36.20</td>
<td>0.26</td>
<td>398.16</td>
</tr>
<tr>
<td>cam_1_16-03-24_04_00_00.mkv</td>
<td>49.20</td>
<td>0.76</td>
<td>272.84</td>
</tr>
<tr>
<td>cam_1_16-03-24_05_00_00.mkv</td>
<td>27.65</td>
<td>0.34</td>
<td>301.52</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv</td>
<td>16.89</td>
<td>0.33</td>
<td>991.96</td>
</tr>
<tr>
<td>cam_1_16-03-24_25_23_00_00.mkv</td>
<td>19.34</td>
<td>0.63</td>
<td>1138.36</td>
</tr>
<tr>
<td>cam_1_16-03-26_00_00_00.mkv</td>
<td>18.64</td>
<td>0.55</td>
<td>368.6</td>
</tr>
<tr>
<td>cam_1_16-03-26_01_00_00.mkv</td>
<td>34.39</td>
<td>0.68</td>
<td>458.09</td>
</tr>
<tr>
<td>cam_1_16-03-26_02_00_00.mkv</td>
<td>15.45</td>
<td>0.23</td>
<td>458.0</td>
</tr>
<tr>
<td>cam_1_16-03-26_03_00_00.mkv</td>
<td>32.36</td>
<td>0.30</td>
<td>301.6</td>
</tr>
<tr>
<td>cam_1_16-03-26_04_00_00.mkv</td>
<td>18.04</td>
<td>0.25</td>
<td>181.4</td>
</tr>
<tr>
<td>cam_1_16-03-26_05_00_00.mkv</td>
<td>19.59</td>
<td>0.19</td>
<td>264.88</td>
</tr>
<tr>
<td>cam_1_16-03-26_06_00_00.mkv</td>
<td>15.66</td>
<td>0.20</td>
<td>384.68</td>
</tr>
<tr>
<td>cam_1_16-04-25_23_00_00.mkv</td>
<td>11.12</td>
<td>0.41</td>
<td>134.16</td>
</tr>
<tr>
<td>cam_1_16-04-26_00_00_00.mkv</td>
<td>39.58</td>
<td>0.45</td>
<td>173.92</td>
</tr>
<tr>
<td>cam_1_16-04-26_01_00_00.mkv</td>
<td>48.86</td>
<td>0.25</td>
<td>40.52</td>
</tr>
<tr>
<td>cam_1_16-04-26_02_00_00.mkv</td>
<td>14.70</td>
<td>0.25</td>
<td>190.72</td>
</tr>
<tr>
<td>cam_1_16-04-26_03_00_00.mkv</td>
<td>29.11</td>
<td>0.36</td>
<td>110.48</td>
</tr>
<tr>
<td>cam_1_16-04-26_04_00_00.mkv</td>
<td>42.81</td>
<td>0.20</td>
<td>68.4</td>
</tr>
<tr>
<td>cam_1_16-04-26_05_00_00.mkv</td>
<td>19.89</td>
<td>0.32</td>
<td>333.6</td>
</tr>
</tbody>
</table>

Table 4.4: Module test showing the performance of the person extraction. Each row shows the average, FN and Error (distance) for each hour. Lastly the sequence length is shown – which shows how much of hour was actually containing persons. In the “Error” column the color represents how good the result is – where the colors are changed between green (good) and red (worse), applying a gradient color ranging from 0 (good) to 1 (worse) respectively.

As it can be seen there is one hour which has an average error distance of 1.02 meters, which mean that the requirement of having an accuracy of 1 m. is not fulfilled for the hour cam_1_16-03-24_01_00_00. Beside this single sample not fulfilling the requirement, cam_1_16-03-22_00_00_00 and cam_1_16-03-24_04_00_00 both have rather high errors. In Table C.1 a table shows every individual sequence, which makes it possible to see which sequences which affects the accuracy the most. The reason for the 3 high samples in this module test is described in the following itemize:

- **cam_1_16-03-22_00_00_00**: (avg. error: 0.90m) The rather high error is caused by an inaccurate annotation of the person walking at far distance from the camera (100 meters). Due to this distance, the error accumulates quickly due to in accurate annotation.
• cam_1_16-03-24_01_00_00: (avg. error: 1.02m) Two persons walks behind some parked cars in the lower part of the image. This causes the algorithm to discard the positions, since the persons are too small according to the distance from the camera. Meanwhile a car drives out of the scene which causes two positions in the cars to be found. Hereby a high displacement between the annotated and algorithm positions are calculated. In Figure 5.31

• cam_1_16-03-24_04_00_00: (avg. 0.76m) The cars parked close to the camera is causing the person positions extraction to fail in 7/72.

Figure 4.50: Algorithm position vs. ground truth for the person’s position. Parked cars causes the positions to be inaccurate.

From this test it is shown that 2) Must be able to detect and recognize individual persons is fulfilled. This is done by splitting BLOBs containing multiple persons both vertically and horizontally. When a person gets partially occluded (horizontally by the pole in the scene or by other persons) their positions are able to be found. If persons are vertically occluded by other people, the system is able to split the persons into two BLOBs, but the position for the upper person, might not be accurate. This indicates that the requirement 6) Must be able to estimate a person’s position when partial occlusion is present is fulfilled (though with a possible inaccurate position for the upper person). The remaining functionality is the tall splitting algorithm, which is able to handle reflections from the pavement, fulfilling 9) Object representation and position should be invariant to reflections caused by rain on the pavement.

Disregarding the three errors described above, persons are able to be extracted with a maximum mean error of 0.68 m. Therefore the 4) Must be able to extract a person’s position with an accuracy of 5.66 m. in area 1, 2.00 m. in area 2 and 1.00 m. in area 3 is considered fulfilled.

4.5 Recap

In general the algorithm work and only one sample caused the overall mean of an hour to be above 1 m. It is observed that there are noise in position extraction, meaning that the position can vary a few pixels between each frame. Therefore the noise seems to be rather normally distributed.

As described in the previous subsections an algorithm has now been developed to:

• Detect persons in the image
• Splitting of wide BLOBs if people walk close to each other
• Splitting of tall BLOBs if people partial occlude each other, by moving at different distances to the camera or puddle causing reflections of the person
• Tall BLOBs created by puddles will be split by the tall split algorithm and the lower part of the BLOB (the puddle) will be removed

To sum up the different steps implemented the following flow chart has been created.

Figure 4.51: Showing the area which will be considered in the test. Positions within the “red” boundaries will be considered. The blue area is the pole, where positions in this area will not be considered in the test.
5. Tracking

The purpose of this module is to associate the outputted, discrete positions from the “Object Extraction”-module over time. This would attach temporal information to each object’s spatial position. As described in Section 2.3, this consists of two tasks: The target tracking which handles trajectories for each person. If a person is occluded in the scene, the target tracking predicts its position based on previous movement. The track management which assigns incoming, discrete positions to new or existing target trackings. The module must fulfill the requirements set in Section 3.3:

4) Must be able to estimate a person’s position with an accuracy of 5.66 m. in area 1, 2.00 m. in area 2 and 1.00 m. in area 3
5) Must be able to predict person’s positions when full occlusion is present.
7) Must add temporal information to discrete, spatial information.

5.1 Target Tracking

The purpose of the target tracking is to handle trajectories for each individual object. Several problems may exist when tracking persons in the scene:

1. The target may be occluded and thus the system is unable to measure its position in one more multiple frames.
2. The measured position may contain noise, compared to the actual position of the object.
Chapter 5. Tracking

Figure 5.1: Map of area around camera 1. Red represent areas where occlusion occur. The dots denote the output of the person extraction algorithm (blue) and the circle, ground truth (green). The track is from video cam_1_16-03-23_03_00_00.mkv, between frame 79518 and 80303.

An example of both cases for camera 1 is shown in Figure 5.1. Firstly, the person is occluded in the image, which justify the missing positions. Additionally, the positions vary from the ground truth. The $x_1$ and $x_2$ positions and their deviance from the ground truth is shown in Figure 5.2. Here, it can be seen that the positions are accurate and precise in the $x_2$-direction. In the $x_1$-direction, the positions are less accurate and not precise.

Figure 5.2: Algorithm position vs. ground truth for the person’s position shown in Figure 5.1.

To cope with these changes, two things must be done:

- The noise (precision) must be adjusted towards the ground truth. In terms of Figure 5.2, the inputted positions should be smoothed.
- The object’s position must be predicted in missing frames.

To coat with the previously described problems, it has been chosen to implement a Kalman filter. The Kalman filter is able to estimate past, present and future states of a process [Welch, Bishop]. Prerequisites for the filter are [Bradski, Kaehler]:

1. **The modeled system is linear**: A person’s state in frame $k$ can be described by its state in frame $k - 1$, for instance if the last position and velocity is known. The system can therefore be considered linear.
2. **The noise in the measurements are white and conform to a Gaussian distribution.** The difference between the ground truth (annotated trajectories) and the algorithm positions are shown in Figure 5.3. It is noted that the mean is not entirely centered around 0. This is presumably a result of the camera being angled on the right side of persons (assuming front-facing persons), thus the resulting center is shifted towards the side of the person. The system can therefore not be considered truly white Gaussian, but is fairly close to a mean of 0.

![Figure 5.3: Deviation from ground truth in meters for the x1 and x2 direction.](image)

In general, the Kalman filter estimates the state $x_k$ given a series of measurements. The state is influenced by three types of movements [Bradski, Kaehler][4]:

- **Dynamical movement**, which is derived directly from the previous state.
- **Control motion**, is an effect on the state based on outside forces applied to the system. In general control systems, this could for instance be the impact of applying break force on a robot. In this system, no control inputs are used.
- **Random motion**, which accounts for noise in the system. As described earlier this is expected to be Gaussian.

The state is defined by **Equation 5.1**

$$x_k = F x_{k-1} + B u_k + w_k$$  \hspace{1cm} (5.1)

Where

- $x_k$: is the state estimate vector at time $k$.
- $w_k$: is process noise at time $k$, with a white Gaussian distribution defined as $N(0, Q_k)$.
- $u_k$: is optional control inputs.
- $F$: is the state transition matrix, mapping the previous state $x_{k-1}$ onto the current state $x_k$.
- $B$: the mapping from control inputs $u_k$ onto the current state $x_k$. 


Measurements $z_k$ which influence the state are described by Equation 5.2. These measurements may not be in the same domain as the state variable, why a measurement mapping may take place.

$$z_k = Hx_k + v_k \quad (5.2)$$

Where

- $x_k$: is the state estimate vector at time $k$
- $z_k$: is the measurement vector at time $k$
- $v_k$: is measurement noise at time $k$, with a white Gaussian distribution defined as $\mathcal{N}(0, R_k)$.
- $H$: is the measurement matrix, mapping the state $x$ onto the measurement $z$.

The true state of the system $x_k$ can not be directly derived. Instead, the Kalman filter tries to estimate the state $\hat{x}_k$ using the known model and its associated Gaussian noise $Q_k$ and $R_k$. This is divided into two steps: prediction and update.

Firstly, the prediction step is calculated, here shown in order of succession. Equation 5.3 calculates the predicted state $\hat{x}_k$ at timestep $k$, given the previously known state $x_{k-1}$. Equation 5.4 determines the error covariance $\hat{P}_k$ at timestep $k$.

$$\hat{x}_k = F_k x_{k-1} + Bu_k \quad (5.3)$$

$$\hat{P}_k = FP_{k-1}F^T + Q_k \quad (5.4)$$

Secondly, the measurement update is calculated. Again, shown in order of succession. Equation 5.5 determines the state $x_k$ based on the difference between the predicted state and the measurement $y$. If $y$ is small, the prediction is considered correct. Contrarily, if $y$ is large, the predicted state is corrected by the Kalman gain $K$ to yield a better estimate of the state $x_k$. The Kalman gain is largely influenced by the measurement’s variance. A large variance yields a smaller gain. In Equation 5.6 the error covariance $P_k$ is updated against the predicted covariance $\hat{P}_k$.

$$x_k = \hat{x}_{k-1} + K_k y \quad (5.5)$$

Where

- $y$: is the difference between the predicted state and the measurement $y = z_k - H_k \hat{x}_{k-1}$.
- $K_k$: is the Kalman gain, defined as $K_k = \hat{P}_k + H_k^T (H_k \hat{P}_k H_k^T + R_k)^{-1}$.

$$P_k = (I - K_k H_k) \hat{P}_k \quad (5.6)$$

An illustration of the prediction step is seen in Figure 5.4.
The tracking must be used to predict a person’s position, when occlusion occurs. Three different cases are identified as the cause of missing positions:

1. If the person becomes a part of the background model from Section 4.1.
2. If the person is occluded by static scene objects, such as signs.
3. If the person is occluded by dynamic scene objects, such as cars or other persons.

In theory, the tracking could predict positions indefinitely. In practice, this could prove problematic as persons may change trajectory during occlusion. A maximum tracker lifespan must therefore be introduced, which ensures consistency.

5.1.1 Design

Derivation of filter parameters

In the scope of “tracking”, the state is defined as person’s position \( x_1 \) and \( x_2 \) and its velocity, \( v_{x1} \) and \( v_{x2} \). Since the positions are based on segmentation and a homography mapping to world coordinates, no known control inputs are used, thus \( Bu \) is neglected and the measured positions can be mapped directly to the state. It is possible to find the velocity, utilizing the constant framerate, but since persons rarely move between pixels in the same rate as frames are updated, this would yield zero velocity regularly and is therefore not measured in \( z \).

\( F \) and \( H \) can change over time, but is here assumed to be constant, due to a constant framerate. The mapping between an object’s previous state \( x_{k-1} \) and its current state \( x_k \) is determined by the relation in Equation 5.7

\[
x_k = x_{k-1} + \frac{v_{k-1}}{fps} \quad v_k = v_{k-1}
\]

Where
fps: is the number of frames per second (25).

$v_{k}$: is the velocity at timestep $k$

The resulting matrices for Equation 5.1-5.7 are given in Equation 5.8

$$x_k = \begin{bmatrix} x_1 \\
                      x_2 \\
                      v_1 \\
                      v_2 \end{bmatrix} \quad F = \begin{bmatrix} 1 & 0 & dt & 0 \\
                      0 & 1 & 0 & dt \\
                      0 & 0 & 1 & 0 \\
                      0 & 0 & 0 & 1 \end{bmatrix} \quad z_k = \begin{bmatrix} z_{x1} \\
                      z_{x2} \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\
                      0 & 1 & 0 & 0 \end{bmatrix}$$ (5.8)

Where

$dt$: is the time elapsed between frames $\frac{1}{fps}$.

**Prediction Lifespan Estimation**

When occlusions are present, the tracker’s positions are estimated from its last known velocity and position. Ideally, the person would follow the same trajectory, which enables the tracker to find the correct person again once the person is clear of the occlusion area. Practically, persons may change paths during occlusion which is difficult to predict. A maximum prediction lifespan is therefore introduced. The choice of selecting a maximum lifespan threshold is a trade-off between creating multiple trackers for the same person (low frame threshold) or predicting person’s positions in missing frames for longer periods of time, which may lead to false positions.

To determine the maximum lifespan of a tracker, the maximum number of consecutive frames where positions are missing (occluded) are analyzed. The training data lifespans are shown in Figure 5.5.

**Figure 5.5:** Number of consecutive occluded frames for the training data.

The occluded positions are shown in Figure 5.6.

From Figure 5.5, one sequence is occluded in 2480 frames (103 s.). From review of the sequence, this is caused by a stationary person in the scene, thus becoming a part of the background. The second largest occlusion sequence is caused by a person walking behind and along the stationary pole for 334 frames (13.3 s.). The remaining occlusions are mainly
5.1. Target Tracking

Figure 5.6: Positions for sequences with sequential frames above 150 frames (top left), 250 frames (top right), 300 frames (bottom left) and all occlusions (bottom right). The videos are captured at 25 FPS.

carried by persons either being occluded by signs for smaller periods of times and walking past the pole for a period of up to 300 frames (12 s).

Two approaches for determining the lifespan are considered:

1. A static threshold defined on the maximum occlusion from Figure 5.6.

2. A dynamical threshold based on known scene properties.

A static threshold would work for each individual scene, but is not a scalable solution when applied to multiple scenes. A dynamical approach is therefore suggested, using knowledge of the segmented positions over time. The hypothesis is, that if no positions are present in an area, then this area is likely to be an occlusion area. The outputted segmented positions for the (single) person sequences are plotted in Figure 5.7. Here, 55 979 positions from persons are outputted. From the mask, occlusions are clearly present around the pole and to a minor extend around the sign.

Figure 5.7: Outputted positions from person extraction module for all single person sequences. A heatmap of the plotted positions (left) and mask of all positions (right) are shown.

The idea is, that if a person’s tracker starts predicting, a search is started in the direction that the person is heading. If any previously known positions are intersected, the prediction lifespan can be extended based on the person’s velocity and distance until intersection. Two cases are identified, which may occur based on the person’s position and heading:
1. An intersection point is found, and the person is estimated to be visible within a calculated timespan.

2. An intersection point is not found and the person is estimated to be invisible indefinitely.

The latter can occur if the person is leaving the scene or the scene is not well modeled as seen near the harbor’s edge in Figure 5.7. This can partially be solved, by applying smoothing around each position, such that individual trajectories merge into one larger area. Before applying smoothing to the image, single pixels wrongly found in the water are eliminated through connected component analysis. From the resulting image, a Gaussian kernel is applied to the image in Figure 5.8.

![Figure 5.8: Effect of applying a Gaussian kernel on each position in Figure 5.7. A Gaussian of 5x5 (left), 10x10 (center) and 15x15 (right) is applied.](image)

Smoothing the image is an alternative which populates the image where positions are assumed to be found in the future. Excessive smoothing removes prediction areas, giving the opposite effect that no occlusion areas are present. It is therefore chosen to smooth with a $5 \times 5$ Gaussian kernel with $\sigma^2 = 2$.

In practice, trackers may also predict in areas known to hold positions, for instance due to occlusions caused by other persons. A minimum prediction threshold is therefore set to 12 seconds based on Figure 5.5.

**Derivation of noise parameters**

The measurement noise $R_k$ and process noise $Q_k$ need to be estimated. These are assumed to be white and conform to a Gaussian distribution. This is done using the single-person sequences, as described in Section 3.5 and shown in Figure 5.3. In the $x_1$-direction $\sigma_1^2 = 2.7978$ and $\mu_1 = 0.2789$. In the $x_2$-direction $\sigma_2^2 = 2.03881$ and $\mu_2 = 0.7533$. The measurement error is set to $R_k = [2.7978, 2.03881]$ on the diagonal and 0 otherwise.

From the training data, it is noted that some errors occur further away due to noise. This could include positions found from reflections in the windows or doors opening as shown in Figure 5.9. Theoretically, this can be handled by the Kalman filter through smoothing. Practically this may not be feasible as noisy positions further away from the tracker could be mixed with other person’s positions. A constraint is therefore added, such that positions are only assigned to the tracker if the Euclidean distance is within 3 standard deviations (99.6%). In the $x_1$ direction $3\sigma = 5.02$ m. and in the $x_1$ direction $3\sigma = 4.28$ m. It is therefore chosen to set the maximum assignment distance to 5 m. Consequently, multiple positions may be found in the same frame. Temporarily, positions are assigned to trackers by smallest distance. If no tracker’s exist within 5 m., a new tracker is created. This is further evaluated in Section 5.2.
5.1. Target Tracking

Figure 5.9: Example of error in outputted positions. Positions are found further away as a consequence of the door being opened. These positions are difficult to handle by the Kalman filter’s smoothing. To ignore these positions, a distance constraint is introduced.

A preliminary test is made on sequences containing only single persons to test which parameters for \( R_k \) and \( Q_k \) yield the lowest variance and mean closest to 0. The test measures the Euclidean distance between the ground truth and the tracker’s position. The position is distinguished between steps where persons are not occluded (prediction/update steps) and occluded positions (prediction-only steps).

<table>
<thead>
<tr>
<th>( Q )</th>
<th>( x_1 ) direction</th>
<th>( x_2 ) direction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prediction-only</td>
<td>Prediction/Update</td>
</tr>
<tr>
<td>( 10^{-1} )</td>
<td>-0.43  6.38  0.22  0.19</td>
<td>-0.06  20.91  0.83  2.41</td>
</tr>
<tr>
<td>( 10^{-2} )</td>
<td>-0.39  6.23  0.22  0.18</td>
<td>-0.18  19.64  0.83  2.35</td>
</tr>
<tr>
<td>( 10^{-3} )</td>
<td>-0.23  5.60  0.23  0.19</td>
<td>-0.11  15.89  0.82  2.38</td>
</tr>
<tr>
<td>( 10^{-4} )</td>
<td>-0.17  5.92  0.24  0.23</td>
<td>0.14   17.02  0.78  2.36</td>
</tr>
<tr>
<td>( 10^{-5} )</td>
<td>-0.17  6.07  0.26  0.32</td>
<td>0.01   16.62  0.70  2.44</td>
</tr>
<tr>
<td>( 10^{-6} )</td>
<td>-0.08  6.39  -0.07 0.40</td>
<td>-0.24  15.88  0.61  2.70</td>
</tr>
<tr>
<td>( 10^{-7} )</td>
<td>-0.07  6.48  0.26  0.52</td>
<td>-0.13  17.69  0.54  2.65</td>
</tr>
<tr>
<td>( 10^{-8} )</td>
<td>-0.05  6.33  0.26  0.51</td>
<td>0.09   19.09  0.51  2.65</td>
</tr>
</tbody>
</table>

Table 5.1: Position error in meters between Kalman filter’s positions (in prediction-only and prediction/update steps) compared to ground truth for the given \( Q \) and \( R \) error parameters, where \( R = [2.3644, 2.7145] \) on the diagonal and zero otherwise. The colors are applied as the absolute value between 0 and the maximal value for each direction (\( x_1 \) or \( x_2 \)) for each variable (\( \mu \) or \( \sigma^2 \)).

The test results are shown in Table 5.1. Several characteristics are observed from inspection of the test results:

1. The tracker’s accuracy is satisfactory from observation of the mean error.
2. The performance in the \( x_2 \) direction is generally poor compared to the \( x_1 \) direction. This is probably a cause of the higher variance seen in Figure 5.3.
3. The error variance increases as state error becomes increasingly smaller.
4. The prediction-only variance is multiple times higher than in the related prediction/update step.
Effects of changing Q values  Firstly, using higher Q values, the variance during prediction/update steps are generally low, whereas the prediction-only step’s variances are higher. From inspection of the outputted trajectories, which are shown in Figure 5.10, this is probably a cause of the tracker adjusting quickly to the segmented positions, which negate the effect of smoothing. As a result, the prediction-only steps perform poorly if the segmented positions vary from the ground truth. Using a lower Q value allows for more smoothing. Consequently, as shown in Figure 5.11 the trajectories are not affected by the segmented positions causing bad estimations of the paths.

Figure 5.10: Tracked path for inputted positions (blue) $Q = 1e^{-1}$ shown in predict/update steps (red) and prediction steps only (yellow). Ground truth is shown in green.

Figure 5.11: Tracked path for inputted positions (blue) $Q = 1e^{-8}$ shown in predict/update steps (red) and prediction steps only (yellow). Ground truth is shown in green.

Reviewing Table 5.1 the overall lowest variance and best value for $Q = 1e^{-3}$. The resulting trajectory for Figure 5.10-5.11 is shown in Figure 5.12.
Creation of multiple trackers  It is observed that segmented positions, which are further away than 5 meters from an existing tracker cause new trackers to appear in 33/44 sequences of the training data. Multiple problems are identified as causes:

a. The prediction speed of the original tracker is not equivalent to that of the person, thus a new tracker is born. The maximum distance constraint thus creates a new tracker, rather than assigning to the predicting tracker. An example is shown in Figure 5.13 between tracker 2-3.

b. The maximum distance constraint set from Figure 5.3 causes new trackers to appear when segmentation noise is present. From inspection of the outputted trajectories these trackers consist of few assigned are positions and thus start predicting when no further points are assigned. An example is shown in Figure 5.12 tracker 3.

c. Positions from the persons far away from the camera cause problems for the tracker in 10/33 sequences. This is probably a cause of the objects being represented by approximately 10-15 pixels in height. If the segmentation algorithm outputs positions with errors of 1-2 pixels, the positions would move greatly in the $x_2$-direction. An example is shown in Figure 5.13 tracker 1. In Section 4.3 a distance constraint for the segmentation algorithm was set such that persons further away than 125 meters are not treated in the segmentation algorithm to prevent these types of errors.

d. A single person is tracked by multiple trackers simultaneously in 2/33 sequences with multiple trackers. From inspection of Figure 5.14 this is probably a combination of a bad prediction as in problem b and noise in the segmentation due to minor occlusion of the person’s feet.

An additional investigation is set up to remove short-lived trackers which are formed as a consequence problem b. Supplementary to the tracker’s maximum prediction lifespan a threshold is set such that trackers composed of few positions assumed to be noise is removed. An example of trackers to be removed is shown in Figure 5.15.

The test is performed on the 31 sequences containing multiple trackers with state noise $Q = 1e^{-3}$. Up to 5 trackers are created for each sequence. The tracker’s total number of assigned positions, identified problems and lifespan are listed in Table 5.2. Table 5.3 summarizes the total number of errors found in the sequences.
Figure 5.13: Example of problems during tracking. Tracker 1 (furthest to right) is created due to small pixel deviations in segmentation at further distances, which causes positions to appear with greater spacing. Tracker 2 follows the trajectory of the actual person, but due to a slow prediction speed compared to the actual person new positions are assigned to tracker 3.

Figure 5.14: Example of problems during tracking. Multiple trackers are created from one person, which is then tracked in parallel.

Figure 5.15: Tracked path for inputted positions (blue) $Q = 1e^{-3}$ shown in predict/update steps (red) and prediction steps only (yellow). Ground truth is shown in green. The trackers to be removed are circled in red.

Case a is difficult to handle as it affected by both the last outputted positions and relies solely on the assumption that the person does not change speed or direction during an occlusion. If the velocity is wrongly found when entering the occlusion area, the maximum lifespan may become large. Potentially it may take up to multiple minutes to traverse the occlusion area, which is up to 30 meters. A limit of 30 seconds is therefore set.

Case b can be handled by assuming that trackers consisting of less than 18 assigned positions are assumed to be noise. As a result, it is chosen to only to allow predictions if trackers contain
5.1. Target Tracking

Table 5.2: Problems found during training, where multiple trackers are created. In all sequences, at most 5 trackers are created. Their associated problems (a-e) and lifespan are shown under each tracker (T1-T5). Trackers denoted with “-” do not occur. Trackers which perform as expected are displayed with a blank space “ ”.

<table>
<thead>
<tr>
<th>Video name</th>
<th>Seq.</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
</tr>
</thead>
<tbody>
<tr>
<td>cam_1_16-03-20_23_00_00</td>
<td>1</td>
<td>967</td>
<td>1</td>
<td>b</td>
<td>16</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>868</td>
<td>22</td>
<td>b</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>584</td>
<td>10</td>
<td>b</td>
<td>2</td>
<td>c</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>916</td>
<td>2</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-20_23_00_00</td>
<td>5</td>
<td>634</td>
<td>2</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-20_23_00_00</td>
<td>6</td>
<td>824</td>
<td>17</td>
<td>b</td>
<td>9</td>
<td>b</td>
</tr>
<tr>
<td>cam_1_16-03-20_23_00_00</td>
<td>7</td>
<td>577</td>
<td>13</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-20_23_00_00</td>
<td>8</td>
<td>736</td>
<td>2</td>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-20_23_00_00</td>
<td>9</td>
<td>1919</td>
<td>118</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-21_04_00_00</td>
<td>1</td>
<td>56</td>
<td>a</td>
<td>667</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-21_04_00_00</td>
<td>2</td>
<td>2056</td>
<td>213</td>
<td>(c)</td>
<td>18</td>
<td>b</td>
</tr>
<tr>
<td>cam_1_16-03-21_05_00_00</td>
<td>3</td>
<td>1481</td>
<td>16</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-21_05_00_00</td>
<td>4</td>
<td>42</td>
<td>ac</td>
<td>1</td>
<td>c</td>
<td>521</td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>1</td>
<td>36</td>
<td>a</td>
<td>667</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>2</td>
<td>824</td>
<td>17</td>
<td>b</td>
<td>9</td>
<td>b</td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>3</td>
<td>577</td>
<td>13</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>4</td>
<td>1481</td>
<td>16</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>5</td>
<td>736</td>
<td>2</td>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>6</td>
<td>1302</td>
<td>17</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>7</td>
<td>42</td>
<td>ac</td>
<td>1</td>
<td>c</td>
<td>521</td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>8</td>
<td>1196</td>
<td>2</td>
<td>b</td>
<td>2</td>
<td>b</td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>9</td>
<td>1138</td>
<td>3</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>10</td>
<td>686</td>
<td>2</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>11</td>
<td>2056</td>
<td>213</td>
<td>(c)</td>
<td>18</td>
<td>b</td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>12</td>
<td>736</td>
<td>2</td>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>13</td>
<td>1302</td>
<td>17</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>14</td>
<td>4</td>
<td>c</td>
<td>887</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>15</td>
<td>1196</td>
<td>2</td>
<td>b</td>
<td>2</td>
<td>b</td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>16</td>
<td>1138</td>
<td>3</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-21_06_00_00</td>
<td>17</td>
<td>1013</td>
<td>59</td>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-22_23_00_00</td>
<td>1</td>
<td>453</td>
<td>d</td>
<td>52</td>
<td>d</td>
<td>1</td>
</tr>
<tr>
<td>cam_1_16-03-22_23_00_00</td>
<td>2</td>
<td>922</td>
<td>1</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-22_23_00_00</td>
<td>3</td>
<td>630</td>
<td>1</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-22_23_00_00</td>
<td>4</td>
<td>428</td>
<td>1</td>
<td>b</td>
<td>1</td>
<td>b</td>
</tr>
<tr>
<td>cam_1_16-03-22_23_00_00</td>
<td>5</td>
<td>1574</td>
<td>3</td>
<td>b</td>
<td>2</td>
<td>b</td>
</tr>
<tr>
<td>cam_1_16-03-23_04_00_00</td>
<td>1</td>
<td>887</td>
<td>77</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-23_04_00_00</td>
<td>2</td>
<td>1018</td>
<td>3</td>
<td>b</td>
<td>12</td>
<td>b</td>
</tr>
<tr>
<td>cam_1_16-03-23_05_00_00</td>
<td>3</td>
<td>509</td>
<td>d</td>
<td>569</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>cam_1_16-03-23_05_00_00</td>
<td>4</td>
<td>1003</td>
<td>2</td>
<td>b</td>
<td>2</td>
<td>b</td>
</tr>
<tr>
<td>cam_1_16-03-23_06_00_00</td>
<td>5</td>
<td>705</td>
<td>1</td>
<td>b</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3: Summarization of tracker problems described in Table 5.2.

More than 18 assigned positions (corresponding to 720 ms @ 25 fps).

Case c can be handled by decreasing the maximum tracker distance in the segmentation algorithm. From visual inspection, errors frequently occur 115 meters away from the camera. A new constraint is therefore set, such that positions found below row 80 in the image is ignored. Case d is difficult to handle, as it appears to be a cause of partial occlusion, causing new trackers to appear in front of the actual target. In multi-target tracking this may resemble two persons walking beside each other.
Chapter 5. Tracking

Reconstructing the results from Table 5.1 with the new constraints yields better performance as shown in Table 5.4. The outputted trajectories, inputted positions and ground truth can be found in Appendix E.

<table>
<thead>
<tr>
<th>Q</th>
<th>( \mu )</th>
<th>( \sigma^2 )</th>
<th>( \mu )</th>
<th>( \sigma^2 )</th>
<th>( \mu )</th>
<th>( \sigma^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 1 \times 10^{-3} ) (old)</td>
<td>-0.23</td>
<td>5.60</td>
<td>0.23</td>
<td>0.19</td>
<td>-0.11</td>
<td>15.89</td>
</tr>
<tr>
<td>( 1 \times 10^{-3} ) (new)</td>
<td>0.23</td>
<td>0.31</td>
<td>0.24</td>
<td>0.14</td>
<td>0.12</td>
<td>3.48</td>
</tr>
</tbody>
</table>

Table 5.4: Performance after adding the constraint, such that trackers created due to noise (case c/d) can be eliminated.

5.1.2 Implementation

The implementation is done as a python module named Tracker. The module contains two classes Tracker, which handles assignment and creation of Kalman trackers and Kalman which handles positions for each individual person. The two classes are shown in Figure 5.16.

```
Tracker
- trackers : List of <Kalman>
+ addPositions(array) : void
+ killOldTrackers() : void
+ predictPositions() : void

Kalman
+ X : array
+ predict() : void
+ z : array
+ update(array) : void
+ age : int
```

Figure 5.16: Class diagram for the tracking module. This is split into two classes: Tracker, which handles each person’s Kalman tracker.

Program Flow

The program is applied on each frame in the video. Firstly, all trackers are evaluated and old or noisy trackers are removed. For remaining trackers their future positions are predicted and distances to incoming positions are calculated. All positions are assigned to trackers, until no more positions remain unassigned. Remaining trackers are assumed to be occluded. The positions for these trackers in the current frame are found from the previous speed and position.

![Program flow](image)

Figure 5.17: Program flow for target tracking module.

Removing Old Trackers

The method removeBadTrackers evaluates the state for each tracker. If their prediction age is above the maximum allowed age, the tracker is assumed to either having left the scene
or have become occluded. Their positions from the last occlusion are removed, as these are assumed to be bad. The tracked positions are saved in a trajectory database for later usage.

Lastly, trackers which did not have any assigned positions in the last frame (age > 0) and have been in less than 18 frames are assumed to be noise. This is illustrated in Figure 5.18.

![Figure 5.18: Flow for removing old or bad trackers.](image)

**Assigning Positions**

The method `addPositions` is divided into two parts: Assigning all positions to existing or new trackers and calculating positions for unassigned (occluded) trackers.

Firstly, positions are evaluated by smallest distance first. If positions are within 5 meters of an unassigned tracker, the position is assigned. If no match is made, a new tracker is initialized at this position with an initial covariance of $P = I$, where $I$ is the identity matrix.

![Figure 5.19: Flow for assigning positions to trackers. If no positions are found within 5 meters, new trackers are created.](image)

Secondly, unassigned trackers are assumed to be occluded. If the tracker had positions assigned in last frame (age = 0), the tracker’s maximum age is determined based on the scene’s occlusion areas. The tracker’s positions in the occlusion area is calculated based on the previous velocity and position and their occlusion age is increased by 1.
Calculating Maximum Lifespan

The maximum lifespan is determined based on the occlusion areas found in Subsection 5.1.2. When a tracker is initially found as being occluded at position $p_{\text{pers,world}}$, the nearest known historical position without occlusion must be found. This requires knowledge of the person’s direction, which should be converted into image coordinates to be compared to the occlusion area mask. This is done in several steps.

Firstly, the intersection with the images’ boundaries in world coordinates is calculated. This requires the boundaries from image coordinates $p_{\text{bound,img}}$ to be mapped into world coordinates $p_{\text{bound,world}}$. The corners of the image’s boundary and their corresponding world coordinates are shown in Table 5.5.

<table>
<thead>
<tr>
<th>Corner</th>
<th>x</th>
<th>y</th>
<th>$x_1$</th>
<th>$x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper left</td>
<td>0</td>
<td>80</td>
<td>23.76</td>
<td>230.60</td>
</tr>
<tr>
<td>Upper right</td>
<td>383</td>
<td>80</td>
<td>-13.97</td>
<td>277.45</td>
</tr>
<tr>
<td>Lower left</td>
<td>0</td>
<td>287</td>
<td>17.37</td>
<td>347.42</td>
</tr>
<tr>
<td>Lower right</td>
<td>383</td>
<td>287</td>
<td>7.87</td>
<td>345.34</td>
</tr>
</tbody>
</table>

Table 5.5: Mapping between image’s boundaries in full width and between row 80 and the bottom row 287.

Four line segments are defined between each edge defined by the corners in world coordinates. A person’s direction towards each of these line segments can be described as a ray with origin at the person’s position and direction $d$ equal to the person’s velocity $d = [v_1, v_2]$. An illustration of the mapping between image and world coordinates are shown in Figure 5.21.
5.1. Target Tracking

Intersection with each of the line segments can thus be described as a ray-line segment intersection test [Ericson6]. Consider a person and a line segment described by the parametric forms in Equation 5.9-5.10 and shown in Figure 5.22:

\[
x_1(t_1) = p_{pers} + dt_1 \quad (5.9)
\]
\[
x_2(t_2) = a + (b - a)t_2 \quad (5.10)
\]

![Figure 5.22: Intersection point \(x_2(t_2)\) between a ray emitting at \(p_{pers}\) and the line segment ab.](image)

In order for persons to collide with the image’s bounds \(t_2 \in [0, 1]\) must hold. Assuming that bounds are convex, this holds for two opposing line segments. An additional constraint is therefore added, such that \(t_1 \in [0, \infty)\), which only finds collisions in the positive direction of the person. \(t_1\) and \(t_2\) are found through use of Equation 5.11. The collision point is then found by insertion of \(t_1\) into Equation 5.9 such that \(p_{bound,world} = x_1(t_1)\):

\[
t_1 = \frac{|v_1 \times v_2|}{v_2 \cdot v_3} \quad t_2 = \frac{v_1 \cdot v_3}{v_2 \cdot v_3} \quad (5.11)
\]

Where

\(v_1 = p_{pers} - a\)
\(v_2 = b - a\)
\(v_3\): is the vector perpendicular to the person \(v_3 = (-d_y, d_x)\)

\(p_{pers,world}\) and \(p_{bound,world}\) is mapped into image space in order to be compared to the occlusion area mask. Here, the pixels on a straight line between \(p_{pers,img}\) and \(p_{bound,img}\). The nearest known non-occlusion pixel \(p_{occl,img}\) is found. This pixel denote the point where persons were lastly known to be. In world coordinates, the maximum lifespan in seconds is calculated.
from the distance to $p_{occl,world}$ from $p_{pers,world}$ divided by the person’s velocity, such that
$s = \frac{|p_{pers,world} - p_{occl,world}|}{v}$. The approach is illustrated in Figure 5.23.

Figure 5.23: Approach for determining the maximum lifespan. Firstly, the person’s intersection with the scene
bounds are found in world coordinates based on the person’s direction. The intersection point and person’s
position is mapped into image space. On a line between these points, the closest known non-occlusion point is
found. This position is mapped into world coordinates. The distance between the nearest non-occlusion point and
person is used to calculate the maximum lifespan using the person’s velocity.

This means that if the $s$ is calculated to be less than 12 seconds, the prediction time is 12
seconds, otherwise the prediction time is set to $s$.

5.2 Track Management

The purpose of the Track management is to manage individual person’s trajectories through
trackers and assigning incoming positions to each of these trackers. This consists of several
tasks:

- Creating new tracker’s when a target enters the scene (target birth).
- Assigning positions to existing trackers, updating their state.
- Signaling a tracker to predict positions when no positions are present, e.g., in case of
  occlusion.
- Removing old trackers when targets leave the scene and are no longer present.
5.2. Track Management

The purpose of the tracking module is to associate the positions from the “Object Extraction” module over time. Each tracker must be able to predict future positions.

The obvious approach is simply to assign incoming positions to existing trackers by smallest distances first. If any trackers do not have positions assigned, this signify that trackers should predict positions. Contrarily, if more positions are available than trackers exist, new persons have entered the scene which need to be handled. Lastly, trackers leaving the scene should be removed.

Two approaches exist for determining which positions to assign to each tracker:

1. **A feature-based approach**: Requires knowledge of the persons being tracked. Assigns positions by its distance between trackers in feature space.

2. **Assignment by distance**: Requires knowledge of the tracker’s previous or future positions. Assigns positions only by its distance between trackers in world coordinate space.

One of the advantages of the feature-based approach, is that the thermal cameras are already invariant to changes in light, compared to traditional RGB cameras. One approach is to utilize the bounding box created around a person, such that features in each bounding box are found, which is compared to a template saved in each tracker.

A disadvantage is, that the resolution of the camera is fairly limited at $384 \times 288$ pixels. As described in Section 2.1 persons only have a width of 8 pixels at the maximal distance of 125 m., which may further complicate feature extraction. An example of a person extracted 125 meters from the camera is shown in Figure 5.25 (left). In order to perform the feature-based approach, the algorithm must be able to identify the same person between frames, or based on historical features accumulated over time. One problem is that persons might change pose, which exposes different heat signatures for the camera which could prove difficult to
compare. An example is shown in Figure 5.25 (middle/right). Additionally, the classifier should be able to distinguish between features of different persons. Consequently, if features are not discriminative, persons may be wrongly assigned.

The second approach is to compare and assign positions by their distances to trackers. A disadvantage to this approach is, that it does not directly include known characteristics of the objects, compared to the feature based approach. An advantage is, that the approach is simple and can be combined with the tracker’s estimate of the person’s future position. Additionally, the previously found distance constraint of 5 meters set in Subsection 5.1.1 would narrow the assignment area. It is therefore chosen to use a distance based approach.

Bernardin, Stiefelhagen[3] defines a good tracker as one, which is able to:

- Find the correct number of object in a frame
- Estimate each positions object as precisely as possible
- Track consistently through each sequence

5.2.1 Design

Position assignment

The approach is to assign the positions to trackers continuously, until all positions are assigned or no more trackers are present. For each frame, one of three cases are occurring:

1. **Number of trackers < Number of positions.** Occurs when persons enter the scene (target birth), or if noise is present. The approach is to assign all positions to existing trackers and assign remaining unassigned positions to new trackers.

2. **Number of trackers > Number of positions.** Occurs when persons leave the scene or become occluded. The approach is to assign positions, if any, and predict positions for remaining, unassigned trackers.

3. **Number of trackers = Number of positions.** Occurs when no persons enter/leave the scene or are occluded. The approach is to assign positions to existing trackers.

Due to the distance constraint of 5 meters, the above have exceptions. Consider the example in Figure 5.26. Here, three unassigned trackers (▲) and positions (x) are present, which means that item 3 is in effect. Two persons (A/B) change course to the left, creating P1 and P2. P1 and P2 are within reach of trackers A and B and would logically be assigned here. Tracker A does not have any positions within the distance constraint and would then be assumed to be occluded. A new tracker would therefore spawn at P3’s position.

When persons leave the scene, their trackers should terminate. As described in Section 2.5, the scene contains static entry/exit areas at the image’s edges or as people become visible from afar. Dynamic entry/exit areas may also occur if persons enter the scene from a car. The static entry/exit areas are shown in Figure 5.27. Trackers are set to terminate in two cases: when their prediction age exceed the maximum lifespan or if persons enter a predefined entry / exit area.
Figure 5.26: Positions are only assigned to trackers if any are found within each tracker’s search area of 5 meters. In this case, P1 and P2 would be assigned between tracker C and B, tracker A would have no assigned positions and turn to prediction, and a new tracker would spawn at P3.

Figure 5.27: Entry exit areas marked in red, as seen from camera 1. The ellipse shows areas where persons become visible from afar.

The approach to assigning positions which are reachable from multiple trackers has an impact on tracking. The straightforward approach would be to calculate distances between each possible tracker-position combination as temporarily done in Subsection 5.1.1. The shortest distances are assumed to be the most correct for a tracker. In a prioritized manner, the approach would be:

1. Calculate distances between all possible tracker-position combinations.
2. For each distance, shortest first:
   (a) Assign to closest tracker
   (b) Mark tracker and position as assigned
   (c) Continue until all trackers or positions are assigned.

A problem would occur with this approach. Consider an elaborated example of Figure 5.26 as shown in Figure 5.28 over the course of three frames. Firstly, the distances between tracker’s and positions are calculated. Assigning shortest-distance first, tracker C would be assigned
to P2. Afterwards, P2 and C would become unavailable. Resultingly, tracker B would be assigned to P1 for a total distance assignment of 2.80 m. In the next frame, the trackers would have crossed paths, causing a wrong tracking of the persons.

The task of assigning positions to trackers is a General Assignment Problem. The previous method is a greedy approach, where trackers are assigned by order of desirability. Instead, it is chosen to consider an approach, where the overall sum of errors is minimized. In Figure 5.29, assigning tracker B to P2 and tracker A to P1 would achieve an overall distance of 2.63 m. Resultingly, trackers would change persons during tracking, causing incorrect tracking.

Figure 5.28: Illustration of the greedy assignment by shortest distance first with distances shown in the table. The persons have turned slightly left, but tracker’s cross path rather than following the persons.

<table>
<thead>
<tr>
<th>Tracker</th>
<th>Predicted position</th>
<th>Incoming position</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1.80</td>
<td>1.12</td>
</tr>
<tr>
<td>P2</td>
<td>1.41</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Figure 5.29: Illustration of the assignment which minimizes the overall distance during assignment. The persons have turned left, which the trackers are able to register.
5.2. Track Management

5.2.2 Implementation

The Tracker class implemented in Subsection 5.1.2 is expanded to include the improved assignment approach. Resultingly, Figure 5.19 is changed, such that positions are no longer assigned by the nearest distance. Rather, positions are evaluated and assigned through use of the Hungarian algorithm \(\text{Kuhn}^2\), which is a solution to the General Assignment Problem minimizing the sum of errors.

Removing Out of Bounds Trackers

removeBadTrackers in Figure 5.18 is expanded to remove trackers, when their positions are outside the scene. The new flow of the program is shown in Figure 5.30.

![Flowchart](image)

**Figure 5.30:** Improved Flow for removing old, bad or out-of-bounds trackers.

In order to determine if persons are outside of the scene, an approach similar to calculating the maximum lifespan is used. Again, using the image’s bounds and person’s positions and velocity the intersection with the boundaries can be found. As shown in Table 5.5 persons may be within the bounds or outside of the bounds.

![Intersection diagram](image)

**Figure 5.31:** Intersection with boundaries described as line segments. The person may both be within the image’s bounds (left) and outside the image’s bounds (right).

When persons are inside the bounds the \(t_1\)’s for the two opposing line segments are positive and negative respectively. If a person has left the scene, both of these terms are negative, such that \(t_1 \in [0, -\infty]\).
5.3 Module Test

The purpose of the test is to satisfy the following requirements:

4 Must be able to estimate a person’s position with an accuracy of 5.66 m. in area 1, 2.00 m. in area 2 and 1.00 m. in area 3
5 Must be able to predict person’s positions when full occlusion is present.
7 Must add temporal information to discrete, spatial information.

The tracker’s accuracy is measured by comparing the tracker’s positions to their associated person’s ground truth. During tracking, three error scenarios may occur as described in Section 5.2.

- **Mismatch**: A tracker may be assigned to multiple persons. Occurs if the algorithm fails to assign to the correct person or switches during tracking.
- **Miss/FN**: No trackers are assigned to an annotated person. Occurs during occlusion, or if a tracker’s age is below 18 frames (assumed to be noise).
- **FP**: A tracker is created, which does not track any annotated persons. Occurs due to noise in segmentation.

To determine the tracker’s performance, it is chosen to use the CLEAR MOT metric \[\text{Bernardin, Stiefelhagen}\]. CLEAR MOT uses two metrics to describe a tracker’s performance, namely the *multiple object tracking precision* (MOTP) and *multiple object tracking accuracy* (MOTA).

The MOTP score describes the total error distance \(d\) between tracked positions and ground truth, averaged over total matches \(c\). The equation is shown in Equation 5.12. The MOTP score demonstrates a tracker’s ability to maintain consistent trajectories independent of errors such as mismatches or misses.

\[
\text{MOTP} = \frac{\sum_{t} d_{t}}{\sum_{t} c_{t}}
\]

Where

\(d_{t}\): is the distance between ground truth and tracker’s position at time \(t\).

\(c_{t}\): is the total number of matches at time \(t\).

The MOTA score describes the tracker’s total error rate \(E_{tot}\), given by the number of misses, false positives and mismatches. The resulting accuracy is then described by \(1-E_{tot}\) as seen in Equation 5.13.

\[
\text{MOTA} = 1 - \frac{\sum_{t}(m_{t} + f_{p_{t}} + mme_{t})}{\sum_{t} g_{t}}
\]

Where

\(m_{t}\): is the number of misses at time \(t\).
5.3. Module Test

\( f_{p_t} \): is the number of false positives at time \( t \).

\( m_{me} \): is the number of mismatches at time \( t \).

\( g_t \): is the total number of persons (ground truth) at time \( t \).

The CLEAR MOT algorithm keeps track of each tracker-ground truth mapping by use of unique tracker and ground truth ids. A threshold \( T = 5 \text{ mm.} \) is introduced as a search area for each tracker-ground truth mapping. For each frame, the following procedure is followed [Bernardin, Stiefelhagen]:

1. For all tracker-ground truth mapping, check and pair if the match is still available within the threshold \( T \). Store distance \( d_t \) and increment \( c_t \) accordingly.

2. For remaining trackers which have not been paired, pair with available ground truth within the threshold, such that the overall sum of distances \( d \) are minimized. Store new tracker-ground truth mapping, distances \( d_t \) and increment \( c_t \) accordingly.

3. If any remaining trackers which do not have any associated ground truth, increment number of FP \( f_{p_t} \). For any remaining ground truth which do not have any associated tracker, increment number of misses \( m_t \).

Procedure

1. Run test_scripts/main_tracker_extractPositions.py to extract the positions from the test data sequences.

2. Run test_scripts/module_tracker_evaluate.py which prepares files for the CLEAR MOT test

3. Set Zone in the script (1,2,3 or 23)

4. Run CLEAR_MOT/tracker_score.py to find all\(^1\):
   - False Positives
   - False Negatives (Miss)
   - Mismatches
   - MOTA (accuracy)
   - MOTP (precision)

Results

The test results including number of annotated persons and trackers are shown in Table 5.6.

<table>
<thead>
<tr>
<th>Zone</th>
<th>MOTP [cm]</th>
<th>Miss-rate [%]</th>
<th>FP [%]</th>
<th>Mismatches [#]</th>
<th>MOTA [%]</th>
<th>Persons</th>
<th>Trackers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zone 3</td>
<td>49</td>
<td>53.2</td>
<td>24.4</td>
<td>166</td>
<td>22.3</td>
<td>295</td>
<td>325</td>
</tr>
<tr>
<td>Zone 2</td>
<td>83</td>
<td>27.4</td>
<td>13.8</td>
<td>136</td>
<td>58.6</td>
<td>249</td>
<td>283</td>
</tr>
<tr>
<td>Zone 1</td>
<td>140</td>
<td>25.4</td>
<td>9.2</td>
<td>324</td>
<td>65.3</td>
<td>280</td>
<td>666</td>
</tr>
</tbody>
</table>

\(^1\)A Perl library is used from https://github.com/Videmo/pymot

Table 5.6: Test results for all videos in each individual zone as defined in Table 3.1
Zone 3 (Center zone) - T=1.00m  The results for zone 3 defined between row 142-287 is shown in Appendix D.

As seen in Table 5.6 the results show that the tracker’s accuracy is low. Only 7/30 videos with MOTA > 50 %. This indicates that the tracker is unable to track within T=1.00 m optimally. Comparing the video’s miss-rate and number of FPs, the errors are within the same range (<20 % difference) in 13/24 videos. The cause is that trackers are removed when $d > T$ creating a FP error whilst incrementing a miss-rate error.

From visual inspection of the output, a common pattern is that trackers are barely outside 1 m. of the annotated position. Two causes are identified, which frequently occur. Firstly, when persons enter the scene, trackers are created which uses a up to two seconds to adjust the velocity of the person. An example is shown in Figure 5.32. Secondly, persons often change trajectory in area 3 due to obstructions such as signs. This cause the tracker to deviate outside of 1 m. of the annotated position. An example is shown in Figure 5.33.

Figure 5.32: Example of person entering the scene, which causes a new tracker to appear. In the first two seconds, the tracker calculates the person’s velocity which causes the tracker to fall behind by 1.1 m. from the person’s annotated position.

Figure 5.33: Example of person which changes path during tracking. This causes the tracker to deviate up to 1.4 m. from the person’s annotated position (middle) before tracking within the threshold (right).

As described in the requirements the accuracy for zone 3 is set to 1.00 m. (T=1.00 m). This requirement can only be satisfied for min. 50 % of trackings in 7/30 videos. A new requirement is therefore set, such that the accuracy for zone 3 should follow the requirement for zone 2. A new zone is defined called zone 23 which contains both zone 2+3 with an accuracy requirement of 2.00 m.

Zone 23 (Center and Front Zone) - T=2.00m  The results for zone 23 defined between row 109-287 are shown in Table 5.7.
Table 5.7: Results from test data by comparing annotated positions in each frame to the tracker’s positions. The miss-rate is the number of unassigned positions divided by the total number of positions. The FPs are trackers created due to noise (assumed to be persons) over the total number of trackers. The mismatches describe the number of times trackers change persons. The MOTA is the tracker’s accuracy when tracking, irregardless of persons. The MOTP is the tracker’s deviation from ground truth.

The errors will be described in the following.

- **Mismatch errors** occur when a tracker changes person multiple times while in the scene. Analyzing the 5/30 videos with mismatches >= 15, two main causes of mismatch errors are found:

1. **Couples walking closely**, which cause switches to appear. Generally, this is due to small variations in person’s positions in the image’s vertical direction. Examples of causes are due to wires splitting persons in half, resulting in only the upper part of the BLOB being found. An example is shown in Figure 5.34.
Chapter 5. Tracking

Figure 5.34: Example of mismatches (tracker switches) occurring during tracking due to small position variations in the image’s vertical direction. First off (upper left), T1 is assigned to the left person and T3 is assigned to the right person. During the next second (lower left/right), the left person’s feet position is found approx 3 pixels higher up in the vertical direction, causing a shift of 2.6 meters from T1 in world coordinates. The distance between T3 and the right person is 2.35 m., which causes the trackers to switch places as seen in the last frame (upper right).

Minor mismatch errors in the remaining videos occur if multiple trackers are created for a single person, for instance due to change in direction during occlusion.

2. Larger groups appearing, which change position in the group or is occluded causing positions to appear and disappear regularly. This causes mismatch problems to accumulate quickly. An example is shown in Figure 5.35.

Figure 5.35: Example of mismatch errors occurring when persons occlude each other and switch positions regularly within a group.

- **FP errors** occur when a tracker is not assigned to an annotated person. Analyzing the 13/30 videos with FP > 10%, the main causes are:

  1. Cars suddenly moving in the scene, which have been part of the background for longer periods of time. If the car starts moving, a large BLOB is created which is assumed to be multiple persons (such that a BLOB split performed). An example is shown in Figure 5.36.
5.3. Module Test

Figure 5.36: Example of car which have become part of the background (upper left) and produces positions when moved (bottom left/right). The positions are consistently found for over 18 consecutive frames, which creates three trackers (upper right).

2. *Irregular behavior during occlusion* caused by persons changing trajectories where no positions are outputted. The tracker thus predicts in the previous direction, causing new trackers to appear when positions are found again. Typically, this is caused by parked cars during weekends. An example is shown in Figure 5.37.

Figure 5.37: Example of occlusions creating new trackers. Persons are tracked moving away from the camera (left), until being occluded by the car and the pole (middle). The persons change trajectory during occlusion which causes new trackers to appear (right).

3. *Reflections in the pavement* cause a wrong split to be found. If the largest convexity defect does not find the optimal splitting point, the lower part of the BLOB will not be classified as a reflection. In this case, when the lower part is compared to the upper part, too much of the water puddle will be above the person, causing the similarity to be low – resulting in an additional position for the water puddle. An example is shown in Figure 5.38.
Figure 5.38: Split point is found too high, resulting the lower BLOB be considered as a person.

4. *Persons Carrying Items* may cause BLOBs to become large. These BLOBs are assumed to be multiple persons requiring to be split. An example is shown in Figure 5.39.

Figure 5.39: Example of person carrying objects, which makes the segmentation algorithm cause a split due to a large BLOB. Since multiple positions are outputted in consecutive frames, trackers appear around the person, causing multiple FP errors to occur.

- **Miss-rate errors** occurs when no trackers are assigned to a person within T.

  1. *Persons occluded by cars*. Occurs frequently during weekends as cars are parked in front of the camera (illegally). The person is partially occluded which classified the person as a non-person. An example is shown in Figure 5.40.

Figure 5.40: Example of a person occluded by a car, which causes a FN error.

- **Miss-rate and FP errors** occurs when trackers are further away from the ground truth. Resultingly, a FP error is counted and the associated person is counted as a miss. This happens in multiple cases, the most common is described underneath:

  1. *Annotation errors* which causes ground truth to become further away from trackers and segmented positions, thus increasing both FP and miss error. Annotation errors
have the largest impact when their deviation is along the vertical image axis. An example is shown in Figure 5.41.

**Figure 5.41**: Example of error between annotated and tracker’s position causing FP errors to occur. The annotated position is offset 4 pixels in the vertical direction (compared visually to the person’s feet position). The resulting Euclidean distance between ground truth and tracker is 2.53 m., which is outside T=2.00 causing an FP error to be accumulated.

2. **Wrong prediction** of persons movement during occlusion. Occurs frequently when persons move between cars or behind the pole and changes trajectory. An example is shown in Figure 5.42.

**Figure 5.42**: Example of person stops during and loses its tracker during occlusion caused by a parked car. The person’s tracker starts predicting, which causes FP and misses to occur.

3. **Persons partially occluded by cars**. The persons feet are occluded by the ground plane, which may cause a person to be found at a greater distance from the annotated position, if any are found. An example is shown in Figure 5.40.

**Figure 5.40**: Example of a person occluded by a car, which causes positions to be found further away from the ground truth, thus counting as a FN and a FP.

**Zone 1 (Upper zone)** The results for zone 1 defined between row 76-108 are shown in Table 5.8.
### Table 5.8: Results from test data by comparing annotated positions in each frame to the tracker’s positions. The miss-rate is the number of unassigned positions divided by the total number of positions. The FPs are trackers created due to noise (assumed to be persons) over the total number of trackers. The mismatches describe the number of times trackers change persons. The MOTA is the trackers accuracy when tracking, irregardless of persons. The MOTP is the tracker’s deviation from ground truth.

The errors will be described in the following.

- **FP errors** FP errors occur less in this area compared to other areas. The two most common errors are presented below.

  1. *Extra trackers are created in or before occlusion areas.* This is mainly caused by two errors. Persons which enter the scene may only have an associated tracker for a few seconds before becoming fully occluded. The tracker’s velocity may therefore not be sufficient, thus creating a new tracker when the person becomes visible again. Secondly, fast-moving objects such as bikes are seen to create multiple trackers, also due to a bad prediction of velocity. An example is shown in Figure 5.44.
5.3. Module Test

Figure 5.44: Example of creation of multiple trackers, both occurring due to a bad estimation of velocity. A bike (left) and a person enter the scene from afar.

- **Miss rate errors**

  - *Trackers are not assigned* before persons have been in the scene for 5-10 meters. Occurs when persons enter the scene from afar. The cause is that small variations in segmentation may cause large shifts in world coordinates. Trackers are thus first created when persons are closer to the camera.

  ![Example of miss rate errors](image)

  **Figure 5.45:** Example of miss rate errors due to persons entering the scene. Due to small variations in pixel value (left) the trackers first appear when the pixels size (in m.) become small enough to be assigned within the 5 m. assignment radius, creating consistent trackers (right).

- **Miss rate and mismatch errors**

  - *Trackers are not assigned* before persons are able to be split fully by the segmentation algorithm due to occlusion at long distances. The trackers may therefore change position internally in the group if only a small part of the group is found. An example is shown in **Figure 5.46**.
Figure 5.46: Example of a group of persons entering the scene from afar (left). Multiple persons are in the group, but only a portion of the persons can be found due to occlusions. The trackers which are created may move around based on which persons are segmented before becoming stable closer to the camera (right).

- **FP and miss rate errors**
  
  - Wrong prediction in occlusion areas due to changing trajectories during occlusion as similar to occlusion caused by cars. Due to the long distance, the person becomes occluded for a long period of time (up to 30 meters). An example is shown in Figure 5.47.

![Figure 5.47: Example of person changing trajectory during occlusion.](image)

**Conclusion** Including zone 23 in Table 5.6, the new results are shown in Table 5.9.

<table>
<thead>
<tr>
<th>Zone 1</th>
<th>MOTP [cm]</th>
<th>Miss rate [%]</th>
<th>FP [%]</th>
<th>Mismatches [#]</th>
<th>MOTA [%]</th>
<th>Persons</th>
<th>Trackers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>140</td>
<td>25.4</td>
<td>9.2</td>
<td>324</td>
<td>65.3</td>
<td>280</td>
<td>666</td>
</tr>
<tr>
<td>Zone 23</td>
<td>80</td>
<td>31.8</td>
<td>13.5</td>
<td>239</td>
<td>54.5</td>
<td>320</td>
<td>388</td>
</tr>
</tbody>
</table>

Table 5.9: Test results for all videos in each individual zone as defined in Table 3.1, where zone 23 is zone 2+3 merged with zone 2’s accuracy requirement of 2.00 m.

The results show, that 65.3 % of zone 1 is able to keep a MOTP within 140 cm. In zone 23, 54.5 % is able to keep a MOTP within 80 cm.

Li et al. [23] defines a tracker’s ability to track persons into three categories based on number of assigned positions (1-miss rate): mostly tracked (>80 %), partially tracked (20-80 %) and mostly lost (<20 %).

<table>
<thead>
<tr>
<th>Zone 1</th>
<th>Mostly tracked</th>
<th>Partially tracked</th>
<th>Mostly lost</th>
<th>Persons</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>84 (30 %)</td>
<td>159 (56 %)</td>
<td>37 (13 %)</td>
<td>280</td>
</tr>
<tr>
<td>Zone 23</td>
<td>171 (53 %)</td>
<td>119 (37 %)</td>
<td>30 (9 %)</td>
<td>320</td>
</tr>
</tbody>
</table>

Table 5.10
The results show that 13 % of persons are mostly lost in zone 1 and 9 % in zone 23. The reasons why persons trajectories are mostly lost is due to persons being occluded (in large groups and behind the pole) and persons occluded by cars.

Comparing Table 5.9 and Table 5.10 Requirement 4) Must be able to estimate a person’s position with an accuracy of 5.66 m. in area 1, 2.00 m. in area 2 and 1.00 m. in area 3. Requirement 5) Must be able to predict person’s positions when full occlusion is present. Requirement 7) Must add temporal information to discrete, spatial information is therefore fulfilled in the majority of trackings.

5.4 Recap

A tracking module has been developed, which is able to:

- Assign persons entering the scene to trackers
- Track individual person’s positions and velocities in 81 % and 87 % of the cases
- Track multiple persons and predict their positions in frames where measured positions are missing in up to 30 seconds
- Delete old trackers when persons leave the scene or is occluded for longer periods of time
- Handle noise which may occur
6. Context Analysis

The purpose of this module is to develop a warning system, such that operators can be warned about possible drowning accidents. This consists of detecting when persons fall into the water and predicting if people are about to fall into the water. The module must fulfill the requirements set in Section 3.3:

10) The system should be able to detect when people fall into the water
11) The number of FN errors must be 0
12) The number of FP errors should be at a minimum
13) Must give warnings 5 seconds before persons are on collision course with the harbor’s edge

6.1 Detecting and Predicting Falls

As described in Section 2.4 two approaches can be used to detect persons falling into the water, such that alarms can be raised:

- Detect harbor edge crossings using a trip-wire
- Detecting the movement during falls using optical flow

An additional approach is implemented, which warns an operator when a person’s trajectory is on collision course with the harbor’s edge:

- Predict potential falls when persons are 5 seconds from colliding with the edge
Detecting Falls

A trip-wire is a virtual line, which can be placed in an image to yield alarms if crossed. As described in Section 2.4 many cameras nowadays have trip-wire functionality. The idea behind the trip-wire is to detect when persons are about to fall in the water. When warned, an operator would be able to observe the actual accident taking place, rather than receiving an alarm when the accident has taken place.

The trip-wire requires activation from moving objects. Utilizing information previously treated in this project, objects are already detected by the background subtraction [Section 4.1] and recognized by the object extraction [Section 4.3]. Using only the background subtraction, all moving objects including birds and dogs are found, which is not desired. Using the object extraction, person are already recognized and represented by positions in world coordinates.

As described in Section 2.4 the trip-wire area is the raised anchoring area defined 60 cm. from the harbor’s edge. Ideally, only persons which fall into the water enter this area. Practically, people enter and leave this area, causing FP’s. From visual inspection of the video, several cases have been found. Figure 6.1 two examples of people who behave dangerously by either playing or peeing close to the harbor edge.

![Figure 6.1: Showing two persons displaying dangerous behavior. Left figure is a person playing at the edge of the harbor. Right figure is a guy standing at the harbor edge, meanwhile he is peeing.](image)

The suggested approach is to extract known feet positions from persons and raise warnings if persons are found within the defined trip-wire area. An example of the flow is shown in Figure 6.2.

![Figure 6.2: Conceptualized approach for detecting persons crossing the harbor’s edge.](image)

As described in Section 2.3 it was observed that a relatively large change in motion is observed when a persons fall into the water. A secondary approach is therefore considered using optical
flow to estimate the motion.

Optical flow is the apparent motion between frames, where the displacements in the image can be found. The concept of finding optical flow, is visualized in Figure 6.3 by a ball moving as a curve through the scene over 3 consecutive frames.

![Figure 6.3: The motion is calculated between two consecutive frames. Lower row shows the calculated flow from the upper frames of the ball.](image)

There are two main ways to find the optical flow either by finding the displacement for all pixel in the image (dense optical flow) or by only looking at a subset of points, represented by features (sparse optical flow). The latter approach finds feature points in the image and match these points between frames. An issue related to using a sparse model is that it is important to find robust features. Resultingly, if features are not robust, persons may fail to be found. The former approach is to use all pixels in the image, without using features. These methods are more sensitive to noise [Nourani-Vatani et al., 28], which may introduce FP’s. Due to the requirement of developing a system where no FN are allowed the dense methods has been chosen.

The basic idea of dense optical flow is to estimate motion in local regions in the image, by estimating the shift between frames. Therefore in order to estimate the motion, pixel intensity is assumed to be constant during the motion estimation [Beauchemin, Barron, 2]. In Figure 6.4 an example of optical flow in 1 dimension is shown.
Chapter 6. Context Analysis

Figure 6.4: The intensity is assumed to be constant, together with a displacement over time. $I_x$ describes the derivative of the intensity in space, and $I_t$ describes the derivative of intensity in time. The is the velocity the signal has been moved in space.

In order to calculate the optical flow, the optical flow constraint needs to be derived. This flow constraint can be defined by the image intensity function which is defined as [Equation 6.1]

$$I(x,t)$$

Which defines an image region $x$ at time $t$. The displacement $\delta$ of the image region can be described by [Equation 6.2]

$$I(x,t) = I(x + \delta x, t + \delta t)$$

By making a Taylor series expansion of [Equation 6.1]

$$I(x,t) = I(x,t) + \Delta I \cdot \delta x + \delta t \cdot I_t + O^2$$

Where

1. order partial derivatives: $\Delta I = (I_x, I_y)$ and $I_t$
2. (and higher order): $O^2$ partial derivatives

The second and higher order derivatives can be removed since they have a very small impact [Beauchemin, Barron 2]. To further simplify Equation 2 $I(x,t)$ can be subtracted on both sides.

$$0 = \nabla I \cdot \delta x + \delta t \cdot I_t$$

By dividing with the change in time $\delta t$, the equation for the optical flow constraint is derived in [Equation 6.1]:

$$0 = \nabla I \cdot v + I_t$$

Where

- Spatial intensity gradient: $\nabla I = (I_x, I_y)$
6.1. Detecting and Predicting Falls

- Image velocity: \( v = (u, v) \) (in x- and y-direction)

As it can be seen, the optical flow constraint is ill-posed since the linear equation has two unknowns. Therefore only the normal vector to \( v \) can be estimated. The normal vector will be pointing in the direction of the local gradient of image intensity as visualized by Figure 6.5 (left):

![Image of optical flow constraint and normal vector](image)

**Figure 6.5:** Left showing the optical flow constraint as a line in velocity space, together with the normal vector showing the motion component of the local gradient.

This also implies that the constraint [Equation 6.1] can be re-written to [Equation 6.6]

\[
v_{\text{normal}} = \frac{-I_t \nabla I}{|\nabla I|} \quad (6.6)
\]

Since [Equation 6.6] only describes the direction of the local gradient the aperture problem occurs. The aperture problem is, that the local gradient might not capture enough information and may therefore be error-prone with respect to the how the velocity is interpreted. The aperture problem is shown in Figure 6.5 (right) where the upper and lower circle show placements on the box where only the local gradient will not be sufficient to describe the velocity.

In order to handle larger motions in the image hierarchical processing can be used [Beauchemin, Barron2]. A well known method is to process the image at different levels of an image pyramid (eg. Gaussian pyramids).

In order to go from local motion gradient to describe the image motion, information from a neighborhood needs to be considered. This can be done by using the approach described by [Farnebäck7]. Here a polynomial expansion is used to make an approximation of the neighborhood around a single pixel by [Equation 6.7]

\[
f(x) \approx x^T \cdot Ax + b^T + c \quad (6.7)
\]

Which comes from the polynomial basis; 1, \( x^2, y^2, x, y, xy \). Assuming that the intensity is constant, and that pixel move coherent within a small region in the image, makes it possible to estimate a displacement for each pixel. In opencv the function `cv2.calcOpticalFlowFarneback()` calculates a two channel image, where the first channel describes the displacement for each pixel in x-direction and the second channel describes the y-direction in Cartesian space. From this displacement the magnitude and direction can be found by [Equation 6.8 and Equation 6.9].

---

Page 113 of 163
\[ I_{\text{MAGNITUDE}} = \sqrt{x(I)^2 + y(I)^2} \quad (6.8) \]

\[ I_{\text{ANGLE}} = \text{atan2}(y(I), x(I)) \quad (6.9) \]

Where the magnitude is described in pixels and the angle in degrees. In order to narrow the features to represent a falling person (and e.g., not a walking person on the bridge) an idea could be to look at the motion direction. In Figure 6.6, the angles are shown with respect to an image:

![Polar coordinate system](image)

**Figure 6.6:** Polar coordinate system showing how the direction found in the image can be related to motion direction.

Where a falling person ideally should have motion directions around 45 – 135 degrees, whereas a walking person on the bridge would be around 0 or 180 degrees.

Different types of motion are identified to occur in the area where the water is located. These types are:

1. **Background noise:** caused by either birds or waves, shown in Figure 6.7 left.

2. **Horizontal motion:** caused by persons walking on the bridge in the bottom left part of the view, shown in Figure 6.7 right.

3. **Fall motion:** caused by persons falling into the water
6.1. Detecting and Predicting Falls

Figure 6.7: Types of FP errors which may occur due to motion. Bird and waves in the water (left) and persons walking horizontally on the bridge (right).

The suggested flow is to detect all motion in frames. If any motion is classified as that belonging to a falling person, an alarm is raised. The approach is shown in Figure 6.8.

![Flow diagram showing the process of detecting optical flow in video frames.](image)

Figure 6.8: Conceptualized approach for detecting optical flow in the video.

Combining methods

Both trip-wire detection and optical flow may include many FP errors due to dangerous behavior, waves, birds and horizontal motion. Focusing on falls caused by persons, optical flow itself could yield FP errors due to noise. Equivalently, trip-wire detection would give FP errors due to persons entering and leaving the area again. A method is proposed which fuses information from optical flow with trip-wire detection. Logically, in order for persons to fall into the water people, must enter the trip-wire zone. To reduce the number of FP errors, the trip-wire must be activated before considering optical flow. Persons cannot reside in both the raised pavement area, while being in a fall above the water. Therefore, a search window must be defined after each trip-wire is triggered, such that the optical flow is activated in the ensuing seconds.

The suggested approach is visualized in Figure 6.9.
Predicting Falls

In order to raise the operator’s awareness when accidents are about to happen, a simple approach is considered using knowledge of person’s current heading from tracking. As described in Subsubsection 2.2, persons are assumed to steer away from the harbor’s edge during nighttime, as persons prefer unobstructed paths home.

A critical line is defined along the water side of the raised anchoring area as described in Subsection 2.4.2 and shown in Figure 6.10.

The Kalman filter predicts a person’s current velocity and position in each frame, based on movement in previous frames. The person’s movement must be predicted towards this line based on the position and velocity, such that warnings are given if persons are on a collision course within 5 seconds.

Figure 6.9: Combined approach which reduces FP errors when detecting drowning accidents.

Figure 6.10: Illustration of the raised anchoring area (red, left) as introduced in Subsection 2.4.2 and the corresponding critical line near the water side (right).
6.1. Design

Trip-wire

The trip-wire is designed by making an area instead of just a single line. As described in subsubsection 6.1, the idea is to use the world position of the persons, since this will make the system more robust to noise (e.g., birds). The trip-wire zone is shown in Figure 6.10. This figure is converted to a binary mask Figure 6.11, which makes it very fast to plot the position in a zero mask and hereby AND the two masks to check if the position is within the trip-wire zone.

\[
\text{inZone} = \text{sum}(\text{PositionMask} \ & \ \text{TripWireMask})
\]  

(6.10)

This trip-wire detection is implemented in the `Tripwire.isInZone()` which uses the world coordinates.

Optical Flow

The idea of the optical flow is to make a classifier which can determine if a person is in a fall. In the dynamic scene where a rather big part of the image shows the water, motion will often be registered in such an area. Since it is desired to detect if a person falls into the water, the idea is to assume that:

1. There will always be some motion in water.
2. A falling person will cause the motion in the water area to be rather different during a fall, due to the sudden increase in speed.

When calculating the dense optical flow, both the velocity and the direction for each pixel will be available. Besides these two values it could also be possible to analyze how many pixels a fall affects. From this the following features should be investigated:

- Pixel moving speed (magnitude)
- Pixel moving direction (angle)
• Pixel which are affected

The amount of pixels which are affected during a fall will vary depending on the size of the person in the image. To avoid having to take into account the size of a person is the approach will be to look at the mean of the features within the blue area [Figure 6.10](left).

In the following an investigation of how the three mentioned features; magnitude, angle and affected pixel area will be carried out. Here the average, magnitude angle and affected pixel sum pixel, will be calculated in order to get an understanding of

• Feature representation of a person walking on the harbor and fall into the water.
• Feature representation of a person walking on the bridge.
• Feature representation of regular noise (captured over a large time frame)

It should be noted that the average of the angle is found in Cartesian space, in order to avoid having the opposite direction cancel out each other.

**Person walking on harbor and falling**
In the following analysis 5 samples have been used to see if there is a correspondence between the average magnitude, average angle and affected pixel sum. The samples have been visually split to determine which frames belongs to 4 different cases, defined as following:

• Case 1: frames **before** a person falls into the water (blue color)
• Case 2: frames **during** a person falls into the water (red color)
• Case 3: frames of **after effect** (splash) of a person falling into the water (brown color)
• Case 4: frames **after** a person falls into the water – swimming in the water (green color)

In the following table distance from the camera to the sample persons is stated:

<table>
<thead>
<tr>
<th>Sample</th>
<th>Distance to camera [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>p0</td>
<td>81.31</td>
</tr>
<tr>
<td>p2</td>
<td>43.47</td>
</tr>
<tr>
<td>p4</td>
<td>41.26</td>
</tr>
<tr>
<td>p6</td>
<td>67.08</td>
</tr>
<tr>
<td>p8</td>
<td>67.92</td>
</tr>
</tbody>
</table>

*Table 6.1*: Showing the start and end frame for each case; Case 1 – before fall, Case 2 – during fall, Case 3 – after effect of fall and Case 4 – after fall. (* this is not considered since the person was out of FOV after fall)

In the following plots the 4 cases are visualized, where all three features are in same scatter plot, and 3 other plots compares the features 1vs1:

**Scatter plot of: magnitude, angle and affected pixel sum**
6.1. Detecting and Predicting Falls

As seen in the 4 plots [Figure 6.12] there is a clear difference in the plots showing that when a person is in the fall (case 2 – red) and the after effect (case 3 – brown) cause a significant increase in magnitude. Some of the case 2 and case 3 points are located around the case 1 and case 4 (optical flow when no persons are falling) – which is because the cases will overlap.

**Person walking on the bridge:**

In the bottom part of the scene a bridge goes from the harbor edge to a restaurant. When people walk on this bridge, their body will cause changes in the area where the optical flow is of interest. 10 samples have therefore been used to see how the optical flow is affected. The samples have been captured from where a person walks towards the bridge and until they leave the bridge. In [Figure 6.13] the results are shown:
Chapter 6. Context Analysis

Figure 6.13: Bridge samples – Upper left: a 3D scatter plot with all the three features. Upper right: the affected pixel sum and magnitude. Bottom left: angle and magnitude. Bottom right: affected pixel sum and magnitude.

Ideally it should be possible to distinguish between bridge walking persons and falling persons by only looking at their motion angle, but from the scatter plots Figure 6.13 this does not seem to be possible.

**Analysis of regular noise:** As earlier described the water will cause motion in the area of interest. Therefore an analysis of regular noise will be performed on three different wind conditions:

- 5 m/s – *cam_1_16-03-21_23_00_00.mkv*
- 8 m/s – *cam_1_16-04-17_23_00_00.mkv*
- 10 m/s – *cam_1_16-04-19_23_00_00.mkv*

To make the analysis all pixels with a magnitude below 1 has been discarded, otherwise the mean of all the pixels will be around 0.27. In the following Figure 6.14–Figure 6.16 the regular noise are shown:
6.1. Detecting and Predicting Falls

5 m/s – cam_1_16-03-21_23_00_00.mkv

Figure 6.14: Scatter plots of the how the noise affect the features.

7.5 m/s – cam_1_16-04-17_23_00_00.mkv

Figure 6.15: Scatter plots of the how the noise affect the features.
Chapter 6. Context Analysis

10 m/s – cam_1_16-04-19_23_00_00.mkv

Figure 6.16: Scatter plots of the how the noise affect the features.

By comparing the plots of **falling persons** and **regular noise** there seems to be a clear pattern showing that the magnitude tells much of what is happening. During the regular noise only 99 samples have a magnitude which surpasses 2.0. This means that during 3 hours 99 FP’s will be triggered. In order to lower this number it is possible to look at the fall motion as a spatiotemporal event. Therefore, investigating the number of consecutive frames where the motion threshold is surpassed should be used as an additional constraint. In Table 6.2 the number of consecutive frames where the training samples are above the threshold are shown:

<table>
<thead>
<tr>
<th>Sample</th>
<th>Consecutive frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>p0</td>
<td>8</td>
</tr>
<tr>
<td>p2</td>
<td>14</td>
</tr>
<tr>
<td>p4</td>
<td>9</td>
</tr>
<tr>
<td>p6</td>
<td>12</td>
</tr>
<tr>
<td>p8</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 6.2: Training samples of persons falling into the water where number of consecutive frames above 2 in magnitude are shown.

As seen in the table the decrease of False-Positives converges around a constraint of 5 consecutive frames. Comparing this constraint with the smallest number of consecutive frames from the persons falling into the water Table 6.2 this constraint will allow a buffer of 3.

By using these two constraints in the optical flow (magnitude > 2 and 5 consecutive frames) this will raise alerts on all the persons walking on the bridge.
6.1. Detecting and Predicting Falls

<table>
<thead>
<tr>
<th>Constraint</th>
<th>False-Positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>45</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 6.3: 3 hours of noise data shows that by using a constraint (left column) the number of consecutive frames decrease rapidly.

![Figure 6.17: Left figure shows the trip-wire zone (red area). If a position is found in this area the trip-wire is activated. Right figure shows the area (blue) where motion should be estimated.](image)

**Prediction**

In order to raise warnings, the critical line is defined as a line segment in world coordinates defined between \( a = [13.32, 341.21] \) and \( b = [14.24, 275.00] \). Again, the problem can be solved using a ray-line segment intersection as in Subsection 5.1.2. Using the person’s velocity in m/s as direction vector and revisiting Equation 5.9, \( t_1 \) denotes the time in seconds, before intersection with the given line. A warning is therefore raised, once \( t_1 \in [0, 5] \).

**6.1.2 Implementation**

To show how the implementation of the fall detection algorithm and the predict fall algorithm the individual flow charts are shown in Figure 6.18 and Figure 6.19.

**Fall Detection**

For each incoming frame the position for each person is found and converted into world coordinates. Using the world position a trip-wire zone check will determine if any persons are located within the critical area. If the trip-wire is activated a counter will be set, stating that the next 125 frames the optical flow needs to be analyzed. If the optical flow is activated.
(finding a person falling) for 5 consecutive frames an alarm is raised.

Figure 6.18: Flow chart describing the implementation of the fall detection system. For each frame the world positions are found. If the trip-wire is activated a counter `remaining frames` will be set to 125 (5 seconds). For the next 5 seconds the optical flow will be analyzed and if the optical flow is activated the counter for consecutive frames will increase. If not the counter will be reset to zero. An alarm will be raised if 5 consecutive frames with optical flow activated is found.

Fall Prediction

In parallel with the above described fall detection algorithm the predict fall algorithm runs. The world position of each person in the image will be assigned to a tracker. For each of the trackers in the current scene the velocity and direction is found. If the person is heading towards the edge of the harbor and will cross it within 5 seconds, a warning is raised.
6.2 Test and recap

The entire system has now been analyzed, described and implemented throughout Chapter 2 – Chapter 6. The system is capable of extracting persons in the scene. By utilizing the extracted foreground information, defining the persons, it is possible to find the feet position of each person. If people are occluding each other or reflections are found in the pavement, these are handled. By having the information in image coordinates the world position accordingly to the camera can be found using the homography mapping. These positions are now used to detect if a person is situated close to the water, by checking if the persons is within the trip-wire zone. Analyzing the motion in the image the number of False-Positives where a person is triggering the trip-wire zone, but not falling in the water, are supposed to be reduced.

So far this makes the system able to detect if a person is falling. To make the system able to predict if a person are in a risk of falling into the water in the near future, a tracking algorithm has been implemented. The tracking makes it possible to connect the person positions over time. Utilizing the information about the velocity and the direction of a person, this is used to raise a warning in case the accident will happen within 5 seconds.

In order to test the system an acceptance test will be carried out in Chapter 7.

---

**Figure 6.19:** For each frame all world positions will be assigned to the running trackers. In each frame the velocity and direction for the tracker is found. If a tracker has a heading towards the edge of the harbor and due to the velocity will cross it within 5 seconds, a warning is raised.
7. Acceptance Testing

The acceptance test will validate the performance of the system – which includes both the fall predictor and the fall detector. In all 100 hours in the time interval 23 - 07 on Thursdays, Fridays and Saturdays in April have been used in the test. The test is divided into the following subtests:

- Fall Detection
- Fall Prediction
- Execution Time

In each subtest, the results are discussed and their requirements are evaluated.

7.1 Fall Detection

The test must satisfy the following requirements:

10) The system should be able to detect when people fall into the water
11) The number of FN errors must be 0
12) The number of FP errors should be at a minimum

In order to do this, the following properties are measured in the test videos:

- **Trip-wire activations**: times the trip-wire has been activated (number of frames).
- **FP**: number of times the system *has raised* an alarm where no person is falling.
- **FN**: number of times the system *has not detected* a person falling into the water.
- **TP**: number of times the system *has detected* a person falling into the water.

Procedure

1. Run `main_accept_test.py`
2. Wait for all videos to finish (99 hours of video in total)
3. The following information will be printed:
Chapter 7. Acceptance Testing

- Total number of alarms
- Total number of trip-wire activations
- Total number of warnings
- Total number of true predictions
- Percentage of true predictions against number of trip-wire activations
- Percentage of true predictions against number of warnings

Results

The test results for the fall detector are shown in Table 7.1. The table shows the following information:

<table>
<thead>
<tr>
<th>Acceptance test data</th>
<th>FP</th>
<th>FN</th>
<th>TP</th>
</tr>
</thead>
<tbody>
<tr>
<td>376</td>
<td>8 (15)</td>
<td>0 / 5</td>
<td>0 / 0</td>
</tr>
<tr>
<td>Acceptance test data (w. falls)</td>
<td>5</td>
<td>0</td>
<td>5 / 5</td>
</tr>
</tbody>
</table>

Table 7.1: Results showing the performance of the fall detector. All the true falls have been found, yielding 0 FN’s. Furthermore it can be seen than, during the 99 hours of video, the trip-wire alarm was activated 376 times and 8 FP’s were found.

As seen in Table 7.1, all test samples containing person falling into the water are detected. This yields 0 FN’s. The numbers in parentheses shows the number of times the alarm has been activated. The reason for having 6 / 5 alarms is, that in test sample p9 the optical flow has been activated two times within the time window of 125 frames after the trip-wire last was activated (meaning that the number of consecutive frames where the magnitude is above threshold have been reached two times). This was caused by the after effect (splash). Requirement 10) The system should be able to detect when people fall into the water and [11] The number of FN errors must be 0 are therefore fulfilled.

In order to identify the problems influencing the system, the errors are narrowed down to the following 5 cases:

a) Person walking on the bridge.

b) Person activates trip-wire, bird in water causes a magnitude above threshold.

c) Person activates trip-wire and throws a bucket of water.

d) Bird flying close to camera causing position to be found at trip wire – flies into area of optical flow.

Table 7.2 shows reason for why the alarm was triggered for each individual sample, together with the frame number where activation happened.

In Figure 7.1–7.3 examples from where the FP alarms are shown.
### Table 7.2:
Indicates where the FPs are found. The horizontal lines separates the samples from each other (the beginning and end activation has happened within 125 frames).

<table>
<thead>
<tr>
<th>Sample</th>
<th>Activated [frame]</th>
<th>Reason</th>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>cam_1_16-04-10_03_00_00.mkv</td>
<td>62404</td>
<td>a</td>
<td>7.1 (upper left)</td>
</tr>
<tr>
<td>cam_1_16-04-16_04_00_00.mkv</td>
<td>68781, 68786</td>
<td>b</td>
<td>7.2 (left)</td>
</tr>
<tr>
<td>cam_1_16-04-16_05_00_00.mkv</td>
<td>89304</td>
<td>a</td>
<td>7.1 (upper right)</td>
</tr>
<tr>
<td>cam_1_16-04-21_23_00_00.mkv</td>
<td>60148</td>
<td>c</td>
<td>7.2 (right)</td>
</tr>
<tr>
<td>cam_1_16-04-22_02_00_00.mkv</td>
<td>84091, 84112, 84130</td>
<td>a</td>
<td>7.1 (lower left)</td>
</tr>
<tr>
<td>cam_1_16-04-22_05_00_00.mkv</td>
<td>86635, 86654, 86687</td>
<td>d</td>
<td>7.3 (left)</td>
</tr>
<tr>
<td>cam_1_16-04-22_05_00_00.mkv</td>
<td>86703</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-04-23_01_00_00.mkv</td>
<td>88462</td>
<td>a</td>
<td>7.1 (lower right)</td>
</tr>
<tr>
<td>cam_1_16-04-23_01_00_00.mkv</td>
<td>88485</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cam_1_16-04-23_05_00_00.mkv</td>
<td>46035</td>
<td>d</td>
<td>7.3 (right)</td>
</tr>
</tbody>
</table>

**Figure 7.1:** FP errors occurring due to persons activating the trip-wire and walking on the bridge (reason a).
Figure 7.2: FP errors occurring due to bird in the water (left, reason b) and person throwing water (right, reason c).

Figure 7.3: FP errors due to birds flying close to the camera activating the trip-wire and the optical flow.

From the results, it is noticeable that a large number of FPs are removed by using the optical flow logic in a combination. If only the trip-wire is the system will yield approximately 3.80 FP alarms / hour – compared to the current implemented will only yield **0.08 FP alarms / hour**. Requirement [12] *The number of FP errors should be at a minimum* is therefore fulfilled.

In Figure 7.4 the TPs are shown.
Figure 7.4: True-Positives are shown. 6/5 alarms were risen, and as it can be seen in last image, this was caused in the same sample within 6 frames.

The training samples of the falling persons can be found in Figure A.5.

7.2 Fall Prediction

The test must satisfy the following requirement:

13) Must give warnings 5 seconds before persons are on collision course with the harbor’s edge

In order to do this, the following properties are measured in the test videos:

- Prediction Warnings: Total number of prediction warnings given where persons are on collision course with the harbor’s edge within 5 seconds.
Chapter 7. Acceptance Testing

- **TP**: Warnings raised where a trip-wire is activated within \( t = 125 \) frames as defined in Equation 7.1.

\[
TP = \sum \left( (t_{trip\_wire} - t_{warning}) < 125 \right) \quad (7.1)
\]

- **FP**: Warnings raised where a trip-wire is not activated within \( t = 125 \) frames.
- **FN**: Trip-wire activations where a warning has not been raised within \( t = 125 \) frames.

**Procedure**

1. Run `main_accept_test.py`
2. Wait for all videos to finish (99 hours of video in total)
3. The following information will be printed:
   - Total number of alarms
   - Total number of trip-wire activations
   - Total number of warnings
   - Total number of true predictions
   - Percentage of true predictions against number of trip-wire activations
   - Percentage of true predictions against number of warnings

### 7.2.1 Results

Results concerning the **fall predictor** is shown in Table 7.3:

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Warnings</th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceptance test data</td>
<td>851</td>
<td>89/376</td>
<td>762/376</td>
<td>287/376</td>
</tr>
</tbody>
</table>

**Table 7.3**: Acceptance test showing results for the fall predictor.

Comparing with the trip-wire warnings, 89/376 (23.67 %) predictions are given within a 5 second frame of persons activating a trip-wire. In Figure 7.5–Figure 7.8 examples of the true predictions are shown, where the left image is the warning and the right where the trip-wire is activated within 5 seconds afterwards.

![Figure 7.5: Warning raised (left), trip-wire activated within 5 seconds after (right).](image_url)
The reason FPs to occur, is firstly due to persons changing trajectory before the harbor, which does not result in a trip-wire activation and cars leaving the scene causing multiple trackers to occur. An example of both cases is shown in Figure 7.9.
In theory, all trip-wire detections should be able to be associated with a warning arisen before the trip-wire was activated. These are mainly caused by persons becoming part of the background, which created a position in the trip-wire zone before moving away. An example is shown in Figure 7.10.

The results show that only 89/851 (10.46 %) of warnings are given before a trip-wire is activated, resulting in warnings. The prediction module alone is therefore not sufficient in predicting if persons would fall into the water. Requirement 13) Must give warnings 5 seconds before persons are on collision course with the harbor’s edge is not fulfilled in the current system.

### 7.2.2 Execution Time

The test must satisfy the following requirement:

- 1) Must be able to process in real-time (processing time must not surpass video length)

The test is run on a computer with the following specifications:

- Asus N501JW
- Ubuntu 14.04 64-bit (Kernel 4.2.0-36-generic)
• CPU: Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz (8-core)
• RAM: 16 GB DDR3-1600 MHz

Procedure

1. Run main_accept_test.py through cProfile
2. Wait for all videos to finish (99 hours of video in total)
3. Find total execution time for entire video and average execution time for:
   • Segmentation Module
   • Tracker Module
   • Fall Detection Module

Results

The results are shown in Table 7.4 and Figure 7.11.

<table>
<thead>
<tr>
<th></th>
<th>Tracker</th>
<th>Trip-Wire</th>
<th>Person Ext.</th>
<th>BG subtraction</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Time [s]</td>
<td>138.45</td>
<td>0.05</td>
<td>138.77</td>
<td>331.90</td>
<td>609.17</td>
</tr>
<tr>
<td>Execution Time [%]</td>
<td>22.72</td>
<td>0.01</td>
<td>22.78</td>
<td>54.48</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 7.4: Average execution time for processing an hour of video.

The program processes 60 minutes of video in 10.15 minutes. The majority of the processing is due to the background subtraction and loading of video (54.48 %). It is noted that the optical flow for detecting falls is 200 % slower than real-time, but is insignificant due to few activations. The results show that the program is able to achieve real-time processing. Requirement [I] Must be able to process in real-time processing time must not surpass video length is therefore fulfilled.

Figure 7.11: Total execution time for analyzing a video.
8. Evaluation

8.1 Conclusion

Preventing unintentional drowning accidents in Denmark has been a complicated problem to solve throughout many years. Since 2008, 8 persons have drowned in Limfjorden, Aalborg. To address this problem a cooperation between Aalborg Kommune, Beredskabscenter Aalborg, Actas, Hikvision and Aalborg University has been started. The origin of this master’s thesis is to carry out initial analysis on how to solve the problem using computer vision techniques.

To develop the system, a dataset has been collected during April and May 2016 at Aalborg’s harbor front. The dataset consists of 155 hours of thermal video recorded at night between 23–07. Of these, 56 hours have been manually annotated, covering a total number of 666 persons.

A preliminary test at Aalborg’s harborfront during March 2016 was carried out. Here, 10 persons volunteered to reenact drowning accidents in 3 °C water. An analysis of the optical flow shows that persons are difficult to track when in the water. The output show that a limited heat signature emitted from persons, due to water covering the body. Persons are therefore difficult to distinguish and track in-water.

The scope of this project is therefore to focus on persons before the actual drowning accident takes place. This forms the basis for the following problem statement:

*How can a system be developed to detect and track persons on the harbor, and warn an operator if the trajectory is abnormal or raise an alarm if a person falls into the water?*

The first step in the system pipeline is to segment the scene into foreground containing moving objects using a Gaussian Mixture Model. When foreground objects are found, persons are extracted and classified from BLOB’s features. This includes properties such as convexity, aspect ratio and height trained known to be persons. This leads to three different subprocesses, which handles splitting of horizontally and vertically occluded persons in BLOBs. Additionally, a logic is implemented which removes reflections from the pavement which occurs during rainy weather. All persons are represented in a world-coordinate plane spatially by their feet’s position.

A Kalman tracking has been implemented, which recursively calculates a person’s velocity and position over time in each frame. Resultingly, trajectories of persons can be formed over time. The trackers are utilized to make a prediction of person’s trajectories in future frames. Dangerous behavior is found if persons are on collision with the harbor’s edge within 5 seconds. This is raised as a warning which can be visually assessed by an operator.

To detect actual drowning accidents, a fall detection algorithm has been developed. This
algorithm utilizes a trip-wire zone which defines a critical area. The person’s spatial information found in each frame is used directly. By only using a trip-wire many false alarms will occur, since many people either walks on the outer edge of harbor during weekends. Therefore in addition to the trip-wire the dense optical flow is analyzed to verify if a person is actually in danger of drowning. This forms the basis for the fall detector.

The warning and alarm system algorithms have been tested using 99 hours of video from all the weekends in April month 2016. This forms the basis for the acceptance test which showed that during 99 hours only 0.08 false positive alarms were raised per hour and all test and training samples of falling persons were captured. The algorithms are able to 1 hour of video in 10.15 minutes, which show real-time performance.

8.2 Discussion and Future work

8.2.1 Kalman Noise Mean

As seen in Figure 5.3 the segmented position’s mean is 0.28 m. in the $x_1$-direction and 0.75 m. in the $x_2$-direction. The Kalman filter assumes that the measurement noise $Q$ is white, Gaussian noise with mean 0. Consequently, the filter may not estimate positions better than the mean Euclidean deviation in both directions of 0.8 m.

8.2.2 Person Extraction Based on Features

First step in order to extract persons is to separate the background from the foreground, this is done using Gaussian Mixture Models. This is a very robust background subtraction method when people moves or at least only stops for a short period of time. In Figure 8.1 an example where a person stops and after approximately 7 seconds starts becoming a part of the background.

![Figure 8.1:](image)

To make the background subtraction part more robust method features could be utilized and hereby avoid using background subtraction. An approach could be to use HOG features as in [Chang et al. 5]. Here, patches containing only single persons are manually extracted in order to train a HOG classifier. At the current state, our system could be utilized to automatically extract these patches containing both single and multiple persons. In doing so, a large dataset of patches containing persons could automatically be collected.
8.2. Discussion and Future work

Separation of multiple persons

The current way BLOBs containing multiple persons are separated, is by analyzing the convexity of the BLOB. As described in the recap of Chapter 4 problems in splitting can occur. Another way to determine if multiple persons are connected in the same BLOB, could be to search for faces in the BLOB where texture is present. Often heads will be different compared to the rest of the body, due to the circularity of the heads as shown in Figure 8.2 (left). The position of the persons could be found by assuming that the lowest vertical pixel before background as a downwards search from the face. The problem when people occlude vertically would though still remain a problem, since the difference between head and torso temperature might not be different as shown in Figure 8.2 (right).

As described in the module test subsection 4.3.2 persons dragging bikes might cause the background subtraction to separate a person into two. It should be considered to implement logic to connect these “unwanted” splits of BLOBs. A possible way to do so could be to look make the bounding box around the BLOBs and check if the overlap significantly with other bounding boxes, as done in Gade, Moeslund [11].

8.2.3 Person Extraction Feedback

The system relies entirely on spatial information. Connecting the person extraction with the tracking could utilize temporal information like height, aspect ratio, multiple persons vs. single persons etc. This would introduce recursive treatment of the information, but could increase the robustness of the system.

8.2.4 Tracking Assignments Based on Features

The tracking algorithm relies only on position information of the person. As seen in Section 5.3 this can cause problems since two trackers might changes owner. In order to avoid these mismatches features could be introduced. A simple way could be to look at the height or aspect ratio of the previous frame where the tracker was updated.

In addition to include features in the Kalman tracking could be to use the Lukas-Kanade Tracker. Here features are matched from frame to frame and are associated. This method
could possibly also be used to substitute the background subtraction, since the motion is found.

### 8.2.5 Keeping Track of Persons in the Scene

Some trajectories/trackers are split during videos caused eg. by the pole placed in the middle of the scene. The current implementation relies entirely on spatial information, but by adding temporal information this could be improved. An approach could be to keep track of the number of persons in the scene together with entry and exits a person have passed. Hereby if a single person walks in the scene and a new tracker appears without having passed an entry point, and another tracker is being lost in a short period of time without exiting the scene, then a logic could connect these two trackers (if the distance is low).

### 8.2.6 Dynamical Occlusion Area Model

The system implemented estimates occlusion areas. This is only performed once but in an final system this could be updated continuously verifying that new static objects in the area have not appeared over time. Hereby the model could be estimated multiple times (over a period) and afterwards these new models could be compared to the previous, in order to validate if such new static objects should be included into the model.

Another thing to have in mind is that the implementation is done in image coordinates and a Gaussian is applied in order to connect positions. This could be considered to be done in world coordinates, since the Gaussian applied in the image coordinates is the same for all points. This means that the Gaussian will have much more affect on the positions at a far distance from the camera in world coordinates, compared to positions close to the camera.

### 8.2.7 Cluttering During Changing Weather

During changes in the weather, cluttering between persons and the pavement might occur. Due to the privacy issues described in Subsection 3.1.3 only the thermal streams are stored. It could be considered to fuse information from the RGB feed in such cases. Based on current information, this should be possible to handle online, if the RGB video is not stored.

### 8.2.8 Detect cars

Currently the system does not handle when cars enter the scene. This causes problems like multiple new positions are found in the car (due to the person extraction implementation), but also when cars park in the scene will cause new “temporarily static objects” in the scene. If persons gets occluded by these car will cause inaccuracy or severe missing positions. Therefore an investigation in detecting cars should be performed. This could be done in several way; finding features like HOG, analyzing BLOB moments etc.

If a dynamical occlusion handling was implemented and updated continuously the system could be able to handle that cars get parked, and if a person walks behind the car, some logic could force the person extraction module not to remove the positions but try to estimate the feet position based on temporal information like the height of the person.
8.2.9 Automatic parameters adjustment

It has been proven that the persons walking at close to the boundary of the the image (125 meters away from camera 1) will be to inaccurate as written in subsubsection 5.1.1, therefore the boundary constraint was changed to 115 meters. To handle such cases could be to automatically change the noise parameters in the implementation of the Kalman tracking. Here the noise parameters $R_k$ and $Q_k$ not matter where in the scene a person is located. It could be argued that a higher noise was allowed when a person a far away from the camera, due to the pixel accuracy decreases at these long distances.

8.2.10 Light hypothesis testing

Utilizing the current system it could be possible to test the hypothesis that the light on the harbor has an impact on how the bypassers behave. The harbor contains several street lamps illuminating it. By comparing the figure (already shown in Section 2.2) many people seem to walk close to the street lamps during the night Figure 8.3 (left). From this it seems that the light have an impact on where people walk during night. This is related to an ongoing debate about the harborfront’s impact on safety as described in [Nordijske26]. Figure 8.3 (right) is shown to give the reader an idea of the area which are being illuminated by the street lamps.

![Heatmap of trajectories in each pixel for 44 persons during the night (left) and street lighting at night (right).](image)

Figure 8.3: Heatmap of trajectories in each pixel for 44 persons during the night (left) and street lighting at night (right). The videos are captured on March 9, 2015.

8.2.11 Tracker Assignment Switches

Many assignment switches occur as described in Section 5.3 This is caused by the position being a few pixel misplaced from the person walking leftmost in the image scene. Due to the camera placement, small offsets in the image’s height cause positions for the leftmost person the appear behind the rightmost person. In Figure 8.4 this is shown. Resultingly, the Hungarian algorithm assigns the leftmost person’s tracker to the rightmost person and vice versa. If the camera were place more perpendicular to the walking path these misplacements would still be assigned to the correct tracker.
8.2.12 Abnormality detection

One of the main purposes of this project is to give a warning if a situation seems to be dangerous, e.g., that a person might fall into the water within a rather short period of time.

In this project a warning method was implemented, which uses the current information about a person's velocity and direction. Combined with the hypothesis that persons do not walk near the edge at night, warnings are raised when person's are on collision course with the harbor's edge within 5 seconds. This hypothesis is based on manually annotated trajectories and through visual inspection. Now that a framework for tracking persons on the harbor is in place, more advanced methods can be considered, which are based on the output of the system. In the follow paragraphs some ideas will be discussed.

Signature-based detection

As described in Subsection 2.4.2, signature-based detection uses templates to detect known types of dangerous behavior. To form a signature-based method it is therefore required to have information about how people behave in given situations. Two approaches are suggested in order to find occurrences of dangerous behavior:

1. Investigate trajectories of persons who are known to be under the influence of alcohol. In order for this to be done, the state of persons in the videos need to be known. A thorough investigation could be done, where persons are asked to use a breathalyser after having completed a trajectory. This would be comprehensive to conduct and care would need to be taken, such that persons are not biased during the test.

2. Link together the samples where the trip-wire is intersected and the trajectory for the person causing this. Hereby, a template of "trip-wire caused" trajectories could be investigated to see if general patterns are formed.

Figure 8.4: Same scene captured from two different views. Misplacements seen from left camera can cause the trackers to switch. If the scene was captured from eg. the right camera these misplacements would just lead to offsets for the trackers, but not switching.
Outlier-based detection

Utilizing known trajectories over time, a model of normal behavior can be made, such that outlier-based detection is feasible. Two approaches could be considered:

1. Finding common properties of trajectories based on via points. An example of an approach is given in [Kalayeh et al. 18]. Here, each trajectory is partitioned into a set of flow vectors. Based on four features x/y position and direction, common points are found through K-means clustering. Points are clustered if they are spatially adjacent and similar in direction. An illustration is shown in Figure 8.5. An extension could be to use other features which can be described quantitatively in new feature dimensions. Features should be chosen, based on whether or not trajectories are impacted. Suggestions include global features such as atmospheric pressure (in hPa), rain (in mm), time of the day, but also in-scene features such as persons nearby or velocity.

The outputted via points could initially be used analytical purposes to give an overview of expected events in the scene. The via point’s clusters could also be utilized. If persons are further away from clusters based on a set threshold warning could be raised at an operator.

2. Evaluating a person’s current trajectory against a known trajectory model. An example would be to divide the scene into a known grid arranged as a Markov Model. In a simple case, each cell would have four probability outputs, based on historical decision from persons who had previously traversed these cells. If the joint probability of the previous cells traversed could be accumulated. If the probability drops below a predefined threshold which is considered abnormal, a warning is raised.

8.2.13 Overlapping cameras

Window Reflections

If the cameras were placed closer to each other the noisy positions caused by the window-reflections could possibly be avoided. Hereby positions could be cross-compared between the two cameras. If positions are not matched in world coordinates, the positions may be due to noise in the camera. This would make the system more prone to noise from the reflections in the windows as shown in Figure 8.6.
Figure 8.6: Reflection are caused by people themselves, therefore by having two cameras capturing the same scene, positions can be compared. Hereby the window reflections can be avoided.

Tracking over both cameras

Currently, the system has only been focused on camera 1. The placement of the cameras should make it possible to track persons over both cameras, which should be possible given that trajectories are given in world coordinates. Supplementary, it would be beneficial to place camera 2 at a higher position, since the current position does not allow it to utilize its full range of 250 m. This is due to occlusions when multiple persons walk at longer distances from the camera. An example is shown in Figure 8.7.

Figure 8.7: Tracking persons in both cameras at the same time. Camera 2 is placed lower than camera 1, which makes persons occlude each other more easily at longer ranges.
Bibliography


A. Trip wire detection

Scatter plot of: magnitude, angle and pixel sum affection

Figure A.1: Upper left: p0 (May), upper right: p2 (May), middle left: p4 (May), middle right: p6 (March), lower left: p8 (March).
Scatter plot of: magnitude and angle

Figure A.2: Upper left: p0 (May), upper right: p2 (May), middle left: p4 (May), middle right: p6 (March), lower left: p8 (March).
Appendix A. Trip wire detection

Scatter plot of: magnitude and pixel sum affection

Figure A.3: Upper left: p0 (May), upper right: p2 (May), middle left: p4 (May), middle right: p6 (March), lower left: p8 (March).
Scatter plot of: angle and pixel sum affection

Figure A.4: Upper left: p0 (May), upper right: p2 (May), middle left: p4 (May), middle right: p6 (March), lower left: p8 (March).
Figure A.5: True-Positives are shown for the training samples. 5/5 alarms were risen, and as it can be seen in last image, this was caused in the same sample within 6 frames.
B. Homography Camera 2

Figure B.1: Overview of area for camera 2. The points denote the light poles used as reference points (lying on the $x_2$ axis). The origo is defined as the position where camera 2 is placed. The map is drawn and measured using http://kortinfo.net

<table>
<thead>
<tr>
<th>i</th>
<th>$p_{i,1}$</th>
<th>$p_{i,2}$</th>
<th>$p_{i,3}$</th>
<th>$p_{i,4}$</th>
<th>$p_{i,5}$</th>
<th>$p_{i,1}$</th>
<th>$p_{i,2}$</th>
<th>$p_{i,3}$</th>
<th>$p_{i,4}$</th>
<th>$p_{i,5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>35.65, 2</td>
<td>35.65, 4</td>
<td>35.65, 6</td>
<td>-</td>
<td>-</td>
<td>145, 286</td>
<td>258, 280</td>
<td>372, 276</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>55.72, 2</td>
<td>55.72, 4</td>
<td>55.72, 6</td>
<td>55.72, 8</td>
<td>-</td>
<td>99, 179</td>
<td>170, 176</td>
<td>244, 172</td>
<td>315, 168</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>76.56, 2</td>
<td>76.56, 4</td>
<td>76.56, 6</td>
<td>76.56, 8</td>
<td>-</td>
<td>78, 128</td>
<td>131, 125</td>
<td>183, 123</td>
<td>237, 121</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>97.44, 2</td>
<td>97.44, 4</td>
<td>97.44, 6</td>
<td>97.44, 8</td>
<td>97.44, 10</td>
<td>64, 97</td>
<td>106, 96</td>
<td>146, 94</td>
<td>187, 93</td>
<td>228, 90</td>
</tr>
<tr>
<td>5</td>
<td>117.41, 2</td>
<td>117.41, 4</td>
<td>117.41, 6</td>
<td>117.41, 8</td>
<td>117.41, 10</td>
<td>56, 76</td>
<td>90, 75</td>
<td>124, 74</td>
<td>158, 73</td>
<td>192, 69</td>
</tr>
<tr>
<td>6</td>
<td>137.43, 2</td>
<td>137.43, 4</td>
<td>137.43, 6</td>
<td>137.43, 8</td>
<td>137.43, 10</td>
<td>49, 63</td>
<td>79, 62</td>
<td>108, 61</td>
<td>137, 59</td>
<td>166, 56</td>
</tr>
<tr>
<td>7</td>
<td>157.40, 2</td>
<td>157.40, 4</td>
<td>157.40, 6</td>
<td>157.40, 8</td>
<td>157.40, 10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table B.1: Reference points for camera 2. The world coordinates are denoted in meters with origo at camera 2’s position. Positions marked with “-” are either occluded or not reachable and not included as reference points.

$$H_{cam2} = \begin{bmatrix} 0.271 & -0.034 & -3.155 \\ 7.383 & -0.245 & 552.481 \\ 0.003 & 0.041 & 1 \end{bmatrix}$$  \hspace{1cm} (B.1)
Figure B.2: Reference positions as viewed in the thermal camera 2. The positions of the feet are used as reference positions.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$p_{i,1}$</th>
<th>$p_{i,2}$</th>
<th>$p_{i,3}$</th>
<th>$p_{i,4}$</th>
<th>$p_{i,5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.23</td>
<td>0.23</td>
<td>0.52</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>0.45</td>
<td>0.36</td>
<td>0.53</td>
<td>0.75</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>0.19</td>
<td>0.21</td>
<td>0.10</td>
<td>0.08</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>0.33</td>
<td>0.89</td>
<td>0.56</td>
<td>1.09</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>1.10</td>
<td>0.73</td>
<td>0.37</td>
<td>0.07</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>0.32</td>
<td>0.50</td>
<td>0.62</td>
<td>0.89</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table B.2: Distance difference between calculated world position and known world position in meters for camera 2. The world positions are calculated from the homography mappings in Equation 4.7.
C. Module Test Person Extraction

<table>
<thead>
<tr>
<th>Sample</th>
<th>FN</th>
<th>Error</th>
<th>Sequence length [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>cam_1_16-03-21_23_00_00.mkv_seq01</td>
<td>21.04</td>
<td>0.29</td>
<td>56.84</td>
</tr>
<tr>
<td>cam_1_16-03-21_23_00_00.mkv_seq02</td>
<td>19.14</td>
<td>0.25</td>
<td>47.24</td>
</tr>
<tr>
<td>cam_1_16-03-21_23_00_00.mkv_seq03</td>
<td>21.63</td>
<td>0.68</td>
<td>49.2</td>
</tr>
<tr>
<td>cam_1_16-03-21_23_00_00.mkv_seq04</td>
<td>19.17</td>
<td>0.46</td>
<td>92.84</td>
</tr>
<tr>
<td>cam_1_16-03-21_23_00_00.mkv_seq05</td>
<td>1.99</td>
<td>0.50</td>
<td>126.68</td>
</tr>
<tr>
<td>cam_1_16-03-21_23_00_00.mkv_seq06</td>
<td>2.12</td>
<td>0.37</td>
<td>26.44</td>
</tr>
<tr>
<td>cam_1_16-03-21_23_00_00.mkv_seq07</td>
<td>0.00</td>
<td>0.62</td>
<td>5.64</td>
</tr>
<tr>
<td>cam_1_16-03-21_23_00_00.mkv_seq08</td>
<td>22.52</td>
<td>0.47</td>
<td>4.44</td>
</tr>
<tr>
<td>cam_1_16-03-21_23_00_00.mkv_seq09</td>
<td>3.46</td>
<td>0.23</td>
<td>27.72</td>
</tr>
<tr>
<td>cam_1_16-03-21_23_00_00.mkv_seq10</td>
<td>1.66</td>
<td>0.22</td>
<td>31.24</td>
</tr>
<tr>
<td>cam_1_16-03-21_23_00_00.mkv_seq11</td>
<td>12.70</td>
<td>0.44</td>
<td>43.16</td>
</tr>
<tr>
<td>cam_1_16-03-22_00_00_00.mkv_seq01</td>
<td>22.53</td>
<td>0.16</td>
<td>43.32</td>
</tr>
<tr>
<td>cam_1_16-03-22_00_00_00.mkv_seq02</td>
<td>35.16</td>
<td>1.34</td>
<td>86.24</td>
</tr>
<tr>
<td>cam_1_16-03-22_01_00_00.mkv_seq01</td>
<td>2.49</td>
<td>0.40</td>
<td>9.64</td>
</tr>
<tr>
<td>cam_1_16-03-22_01_00_00.mkv_seq02</td>
<td>4.54</td>
<td>0.37</td>
<td>17.64</td>
</tr>
<tr>
<td>cam_1_16-03-22_01_00_00.mkv_seq03</td>
<td>11.80</td>
<td>0.28</td>
<td>14.92</td>
</tr>
<tr>
<td>cam_1_16-03-22_01_00_00.mkv_seq04</td>
<td>27.28</td>
<td>0.31</td>
<td>54.84</td>
</tr>
<tr>
<td>cam_1_16-03-22_01_00_00.mkv_seq05</td>
<td>6.74</td>
<td>0.43</td>
<td>58.76</td>
</tr>
<tr>
<td>cam_1_16-03-22_01_00_00.mkv_seq06</td>
<td>24.27</td>
<td>0.22</td>
<td>38.24</td>
</tr>
<tr>
<td>cam_1_16-03-22_01_00_00.mkv_seq07</td>
<td>18.78</td>
<td>0.31</td>
<td>66.44</td>
</tr>
<tr>
<td>cam_1_16-03-22_02_00_00.mkv_seq01</td>
<td>13.57</td>
<td>0.27</td>
<td>65.16</td>
</tr>
<tr>
<td>cam_1_16-03-22_02_00_00.mkv_seq02</td>
<td>22.64</td>
<td>0.32</td>
<td>57.24</td>
</tr>
<tr>
<td>cam_1_16-03-22_02_00_00.mkv_seq03</td>
<td>7.25</td>
<td>0.27</td>
<td>64.04</td>
</tr>
<tr>
<td>cam_1_16-03-22_02_00_00.mkv_seq04</td>
<td>10.30</td>
<td>0.14</td>
<td>21.36</td>
</tr>
<tr>
<td>cam_1_16-03-22_02_00_00.mkv_seq05</td>
<td>25.39</td>
<td>0.29</td>
<td>59.24</td>
</tr>
<tr>
<td>cam_1_16-03-22_02_00_00.mkv_seq06</td>
<td>12.44</td>
<td>0.18</td>
<td>40.84</td>
</tr>
<tr>
<td>cam_1_16-03-22_05_00_00.mkv_seq01</td>
<td>10.50</td>
<td>0.19</td>
<td>67.84</td>
</tr>
<tr>
<td>cam_1_16-03-22_05_00_00.mkv_seq02</td>
<td>17.73</td>
<td>0.22</td>
<td>56.84</td>
</tr>
<tr>
<td>cam_1_16-03-22_06_00_00.mkv_seq01</td>
<td>12.42</td>
<td>0.16</td>
<td>44.44</td>
</tr>
<tr>
<td>cam_1_16-03-22_06_00_00.mkv_seq02</td>
<td>19.24</td>
<td>0.20</td>
<td>64.04</td>
</tr>
<tr>
<td>cam_1_16-03-22_06_00_00.mkv_seq03</td>
<td>81.04</td>
<td>0.19</td>
<td>197.24</td>
</tr>
<tr>
<td>cam_1_16-03-22_06_00_00.mkv_seq04</td>
<td>9.26</td>
<td>0.16</td>
<td>54.44</td>
</tr>
<tr>
<td>cam_1_16-03-22_06_00_00.mkv_seq05</td>
<td>16.42</td>
<td>0.73</td>
<td>98.44</td>
</tr>
<tr>
<td>cam_1_16-03-22_06_00_00.mkv_seq06</td>
<td>20.17</td>
<td>0.11</td>
<td>43.04</td>
</tr>
<tr>
<td>cam_1_16-03-22_06_00_00.mkv_seq07</td>
<td>1.02</td>
<td>0.31</td>
<td>39.32</td>
</tr>
<tr>
<td>cam_1_16-03-22_06_00_00.mkv_seq08</td>
<td>46.18</td>
<td>0.10</td>
<td>52.84</td>
</tr>
<tr>
<td>Filename</td>
<td>Value</td>
<td>Error</td>
<td>Distance</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>cam_1_16-03-22_06_00_00.mkv_seq09</td>
<td>10.22</td>
<td>0.15</td>
<td>52.84</td>
</tr>
<tr>
<td>cam_1_16-03-23_23_00_00.mkv_seq01</td>
<td>13.23</td>
<td>0.18</td>
<td>19.96</td>
</tr>
<tr>
<td>cam_1_16-03-23_23_00_00.mkv_seq02</td>
<td>9.52</td>
<td>0.33</td>
<td>57.96</td>
</tr>
<tr>
<td>cam_1_16-03-23_23_00_00.mkv_seq03</td>
<td>11.58</td>
<td>0.32</td>
<td>67.72</td>
</tr>
<tr>
<td>cam_1_16-03-23_23_00_00.mkv_seq04</td>
<td>5.67</td>
<td>1.34</td>
<td>62.76</td>
</tr>
<tr>
<td>cam_1_16-03-23_23_00_00.mkv_seq05</td>
<td>11.98</td>
<td>0.18</td>
<td>54.44</td>
</tr>
<tr>
<td>cam_1_16-03-23_23_00_00.mkv_seq06</td>
<td>3.14</td>
<td>0.77</td>
<td>62.76</td>
</tr>
<tr>
<td>cam_1_16-03-23_23_00_00.mkv_seq07</td>
<td>8.43</td>
<td>0.21</td>
<td>53.64</td>
</tr>
<tr>
<td>cam_1_16-03-23_23_00_00.mkv_seq08</td>
<td>9.88</td>
<td>0.53</td>
<td>83.84</td>
</tr>
<tr>
<td>cam_1_16-03-23_23_00_00.mkv_seq09</td>
<td>33.60</td>
<td>0.29</td>
<td>49.52</td>
</tr>
<tr>
<td>cam_1_16-03-23_23_00_00.mkv_seq10</td>
<td>3.65</td>
<td>0.39</td>
<td>53.64</td>
</tr>
<tr>
<td>cam_1_16-03-23_23_00_00.mkv_seq11</td>
<td>11.07</td>
<td>0.35</td>
<td>61.04</td>
</tr>
<tr>
<td>cam_1_16-03-23_23_00_00.mkv_seq12</td>
<td>5.91</td>
<td>0.26</td>
<td>37.24</td>
</tr>
<tr>
<td>cam_1_16-03-23_23_00_00.mkv_seq13</td>
<td>12.27</td>
<td>0.25</td>
<td>84.76</td>
</tr>
<tr>
<td>cam_1_16-03-23_23_00_00.mkv_seq14</td>
<td>6.80</td>
<td>0.48</td>
<td>49.44</td>
</tr>
<tr>
<td>cam_1_16-03-24_01_00_00.mkv_seq01</td>
<td>19.91</td>
<td>0.18</td>
<td>9.24</td>
</tr>
<tr>
<td>cam_1_16-03-24_01_00_00.mkv_seq02</td>
<td>10.64</td>
<td>0.23</td>
<td>1.88</td>
</tr>
<tr>
<td>cam_1_16-03-24_01_00_00.mkv_seq03</td>
<td>12.90</td>
<td>0.35</td>
<td>1.24</td>
</tr>
<tr>
<td>cam_1_16-03-24_01_00_00.mkv_seq04</td>
<td>1.73</td>
<td>0.30</td>
<td>113.04</td>
</tr>
<tr>
<td>cam_1_16-03-24_01_00_00.mkv_seq05</td>
<td>4.63</td>
<td>0.20</td>
<td>38.84</td>
</tr>
<tr>
<td>cam_1_16-03-24_01_00_00.mkv_seq06</td>
<td>12.40</td>
<td>0.47</td>
<td>74.84</td>
</tr>
<tr>
<td>cam_1_16-03-24_01_00_00.mkv_seq07</td>
<td>33.36</td>
<td>3.06</td>
<td>182.04</td>
</tr>
<tr>
<td>cam_1_16-03-24_01_00_00.mkv_seq08</td>
<td>11.93</td>
<td>0.35</td>
<td>108.96</td>
</tr>
<tr>
<td>cam_1_16-03-24_01_00_00.mkv_seq09</td>
<td>7.78</td>
<td>0.41</td>
<td>48.84</td>
</tr>
<tr>
<td>cam_1_16-03-24_02_00_00.mkv_seq01</td>
<td>20.25</td>
<td>0.16</td>
<td>60.84</td>
</tr>
<tr>
<td>cam_1_16-03-24_02_00_00.mkv_seq02</td>
<td>24.44</td>
<td>0.13</td>
<td>46.16</td>
</tr>
<tr>
<td>cam_1_16-03-24_02_00_00.mkv_seq03</td>
<td>24.56</td>
<td>0.16</td>
<td>47.24</td>
</tr>
<tr>
<td>cam_1_16-03-24_02_00_00.mkv_seq04</td>
<td>18.65</td>
<td>0.15</td>
<td>43.12</td>
</tr>
<tr>
<td>cam_1_16-03-24_02_00_00.mkv_seq05</td>
<td>53.72</td>
<td>0.10</td>
<td>77.96</td>
</tr>
<tr>
<td>cam_1_16-03-24_02_00_00.mkv_seq06</td>
<td>56.22</td>
<td>0.10</td>
<td>40.84</td>
</tr>
<tr>
<td>cam_1_16-03-24_02_00_00.mkv_seq07</td>
<td>48.34</td>
<td>0.25</td>
<td>105.92</td>
</tr>
<tr>
<td>cam_1_16-03-24_03_00_00.mkv_seq01</td>
<td>18.42</td>
<td>0.57</td>
<td>38.44</td>
</tr>
<tr>
<td>cam_1_16-03-24_03_00_00.mkv_seq02</td>
<td>28.67</td>
<td>0.34</td>
<td>35.44</td>
</tr>
<tr>
<td>cam_1_16-03-24_03_00_00.mkv_seq03</td>
<td>4.96</td>
<td>0.27</td>
<td>56.44</td>
</tr>
<tr>
<td>cam_1_16-03-24_04_00_00.mkv_seq01</td>
<td>20.80</td>
<td>0.13</td>
<td>36.16</td>
</tr>
<tr>
<td>cam_1_16-03-24_04_00_00.mkv_seq02</td>
<td>17.03</td>
<td>0.11</td>
<td>31.0</td>
</tr>
<tr>
<td>cam_1_16-03-24_04_00_00.mkv_seq03</td>
<td>51.06</td>
<td>0.16</td>
<td>5.64</td>
</tr>
<tr>
<td>cam_1_16-03-24_04_00_00.mkv_seq04</td>
<td>67.44</td>
<td>0.13</td>
<td>6.88</td>
</tr>
<tr>
<td>cam_1_16-03-24_04_00_00.mkv_seq05</td>
<td>80.75</td>
<td>0.31</td>
<td>6.44</td>
</tr>
<tr>
<td>cam_1_16-03-24_04_00_00.mkv_seq06</td>
<td>72.77</td>
<td>0.17</td>
<td>7.64</td>
</tr>
<tr>
<td>cam_1_16-03-24_04_00_00.mkv_seq07</td>
<td>83.65</td>
<td>3.97</td>
<td>114.04</td>
</tr>
<tr>
<td>cam_1_16-03-24_04_00_00.mkv_seq08</td>
<td>2.74</td>
<td>0.80</td>
<td>26.28</td>
</tr>
<tr>
<td>cam_1_16-03-24_04_00_00.mkv_seq09</td>
<td>18.16</td>
<td>0.11</td>
<td>38.76</td>
</tr>
<tr>
<td>cam_1_16-03-24_05_00_00.mkv_seq01</td>
<td>7.54</td>
<td>0.50</td>
<td>44.04</td>
</tr>
<tr>
<td>cam_1_16-03-24_05_00_00.mkv_seq02</td>
<td>2.34</td>
<td>0.51</td>
<td>61.56</td>
</tr>
<tr>
<td>cam_1_16-03-24_05_00_00.mkv_seq03</td>
<td>53.70</td>
<td>0.18</td>
<td>132.44</td>
</tr>
<tr>
<td>cam_1_16-03-24_05_00_00.mkv_seq04</td>
<td>11.78</td>
<td>0.23</td>
<td>63.48</td>
</tr>
<tr>
<td>Video Name</td>
<td>Size</td>
<td>Duration</td>
<td>File Size</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------</td>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv_seq01</td>
<td>16.79</td>
<td>0.10</td>
<td>38.84</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv_seq02</td>
<td>35.48</td>
<td>0.14</td>
<td>1.24</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv_seq03</td>
<td>6.10</td>
<td>0.36</td>
<td>147.44</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv_seq04</td>
<td>16.94</td>
<td>0.41</td>
<td>113.12</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv_seq05</td>
<td>5.63</td>
<td>0.68</td>
<td>100.84</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv_seq06</td>
<td>8.29</td>
<td>0.29</td>
<td>38.6</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv_seq07</td>
<td>38.45</td>
<td>0.18</td>
<td>135.24</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv_seq08</td>
<td>37.01</td>
<td>0.23</td>
<td>11.24</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv_seq09</td>
<td>3.33</td>
<td>0.33</td>
<td>45.64</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv_seq10</td>
<td>20.10</td>
<td>0.19</td>
<td>74.84</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv_seq11</td>
<td>6.60</td>
<td>0.51</td>
<td>43.64</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv_seq12</td>
<td>0.60</td>
<td>0.20</td>
<td>26.84</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv_seq13</td>
<td>24.49</td>
<td>0.16</td>
<td>53.24</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv_seq14</td>
<td>29.08</td>
<td>0.11</td>
<td>27.92</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv_seq15</td>
<td>22.63</td>
<td>0.24</td>
<td>109.24</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00.mkv_seq16</td>
<td>7.99</td>
<td>0.35</td>
<td>24.04</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq01</td>
<td>9.31</td>
<td>0.12</td>
<td>67.48</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq02</td>
<td>0.00</td>
<td>0.29</td>
<td>64.04</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq03</td>
<td>23.88</td>
<td>0.31</td>
<td>15.24</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq04</td>
<td>13.18</td>
<td>1.29</td>
<td>158.44</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq05</td>
<td>16.51</td>
<td>0.39</td>
<td>12.84</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq06</td>
<td>18.56</td>
<td>0.25</td>
<td>11.64</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq07</td>
<td>21.59</td>
<td>0.40</td>
<td>3.52</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq08</td>
<td>18.31</td>
<td>0.29</td>
<td>2.84</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq09</td>
<td>68.10</td>
<td>0.26</td>
<td>43.64</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq10</td>
<td>2.72</td>
<td>0.17</td>
<td>17.64</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq11</td>
<td>1.66</td>
<td>0.50</td>
<td>62.72</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq12</td>
<td>24.24</td>
<td>0.29</td>
<td>80.04</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq13</td>
<td>30.19</td>
<td>0.79</td>
<td>217.68</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq14</td>
<td>24.23</td>
<td>0.93</td>
<td>68.84</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq15</td>
<td>39.92</td>
<td>0.98</td>
<td>72.44</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq16</td>
<td>3.56</td>
<td>0.23</td>
<td>11.24</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq17</td>
<td>8.05</td>
<td>0.31</td>
<td>75.56</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq18</td>
<td>4.80</td>
<td>0.18</td>
<td>61.64</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq19</td>
<td>11.87</td>
<td>0.70</td>
<td>74.84</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00.mkv_seq20</td>
<td>21.95</td>
<td>1.52</td>
<td>16.04</td>
</tr>
<tr>
<td>cam_1_16-03-26_00_00_00.mkv_seq01</td>
<td>67.64</td>
<td>1.71</td>
<td>13.72</td>
</tr>
<tr>
<td>cam_1_16-03-26_00_00_00.mkv_seq02</td>
<td>2.45</td>
<td>0.19</td>
<td>21.24</td>
</tr>
<tr>
<td>cam_1_16-03-26_00_00_00.mkv_seq03</td>
<td>26.85</td>
<td>0.16</td>
<td>36.2</td>
</tr>
<tr>
<td>cam_1_16-03-26_00_00_00.mkv_seq04</td>
<td>7.83</td>
<td>0.18</td>
<td>69.96</td>
</tr>
<tr>
<td>cam_1_16-03-26_00_00_00.mkv_seq05</td>
<td>21.97</td>
<td>0.19</td>
<td>63.36</td>
</tr>
<tr>
<td>cam_1_16-03-26_00_00_00.mkv_seq06</td>
<td>58.73</td>
<td>0.42</td>
<td>14.44</td>
</tr>
<tr>
<td>cam_1_16-03-26_00_00_00.mkv_seq07</td>
<td>10.00</td>
<td>1.00</td>
<td>140.44</td>
</tr>
<tr>
<td>cam_1_16-03-26_01_00_00.mkv_seq01</td>
<td>57.50</td>
<td>0.18</td>
<td>71.16</td>
</tr>
<tr>
<td>cam_1_16-03-26_01_00_00.mkv_seq02</td>
<td>76.33</td>
<td>1.26</td>
<td>8.28</td>
</tr>
<tr>
<td>cam_1_16-03-26_01_00_00.mkv_seq03</td>
<td>46.03</td>
<td>0.15</td>
<td>46.84</td>
</tr>
<tr>
<td>Path</td>
<td>Code</td>
<td>Accuracy</td>
<td>Confidence</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>----------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>cam_1_16-03-26_01_00_00.mkv_seq04</td>
<td>95.12</td>
<td>0.09</td>
<td>1.64</td>
</tr>
<tr>
<td>cam_1_16-03-26_01_00_00.mkv_seq05</td>
<td>27.02</td>
<td>0.14</td>
<td>54.92</td>
</tr>
<tr>
<td>cam_1_16-03-26_01_00_00.mkv_seq06</td>
<td>54.24</td>
<td>0.53</td>
<td>10.84</td>
</tr>
<tr>
<td>cam_1_16-03-26_01_00_00.mkv_seq07</td>
<td>7.19</td>
<td>3.42</td>
<td>41.72</td>
</tr>
<tr>
<td>cam_1_16-03-26_01_00_00.mkv_seq08</td>
<td>59.65</td>
<td>0.50</td>
<td>6.84</td>
</tr>
<tr>
<td>cam_1_16-03-26_01_00_00.mkv_seq09</td>
<td>67.01</td>
<td>0.92</td>
<td>19.52</td>
</tr>
<tr>
<td>cam_1_16-03-26_01_00_00.mkv_seq10</td>
<td>4.87</td>
<td>0.35</td>
<td>32.04</td>
</tr>
<tr>
<td>cam_1_16-03-26_01_00_00.mkv_seq11</td>
<td>56.51</td>
<td>0.28</td>
<td>40.84</td>
</tr>
<tr>
<td>cam_1_16-03-26_01_00_00.mkv_seq12</td>
<td>17.86</td>
<td>0.17</td>
<td>50.84</td>
</tr>
<tr>
<td>cam_1_16-03-26_01_00_00.mkv_seq13</td>
<td>12.18</td>
<td>0.53</td>
<td>17.08</td>
</tr>
<tr>
<td>cam_1_16-03-26_01_00_00.mkv_seq14</td>
<td>18.89</td>
<td>0.32</td>
<td>55.48</td>
</tr>
<tr>
<td>cam_1_16-03-26_02_00_00.mkv_seq01</td>
<td>2.28</td>
<td>0.14</td>
<td>31.64</td>
</tr>
<tr>
<td>cam_1_16-03-26_02_00_00.mkv_seq02</td>
<td>10.71</td>
<td>0.30</td>
<td>59.36</td>
</tr>
<tr>
<td>cam_1_16-03-26_02_00_00.mkv_seq03</td>
<td>38.74</td>
<td>1.37</td>
<td>7.64</td>
</tr>
<tr>
<td>cam_1_16-03-26_02_00_00.mkv_seq04</td>
<td>4.63</td>
<td>0.15</td>
<td>38.04</td>
</tr>
<tr>
<td>cam_1_16-03-26_02_00_00.mkv_seq05</td>
<td>13.44</td>
<td>0.16</td>
<td>45.24</td>
</tr>
<tr>
<td>cam_1_16-03-26_02_00_00.mkv_seq06</td>
<td>12.92</td>
<td>0.17</td>
<td>70.28</td>
</tr>
<tr>
<td>cam_1_16-03-26_02_00_00.mkv_seq07</td>
<td>16.74</td>
<td>0.29</td>
<td>43.24</td>
</tr>
<tr>
<td>cam_1_16-03-26_02_00_00.mkv_seq08</td>
<td>0.00</td>
<td>0.17</td>
<td>0.04</td>
</tr>
<tr>
<td>cam_1_16-03-26_02_00_00.mkv_seq09</td>
<td>0.00</td>
<td>0.18</td>
<td>4.64</td>
</tr>
<tr>
<td>cam_1_16-03-26_02_00_00.mkv_seq10</td>
<td>25.93</td>
<td>0.25</td>
<td>112.44</td>
</tr>
<tr>
<td>cam_1_16-03-26_02_00_00.mkv_seq11</td>
<td>16.29</td>
<td>0.22</td>
<td>45.44</td>
</tr>
<tr>
<td>cam_1_16-03-26_03_00_00.mkv_seq01</td>
<td>52.38</td>
<td>0.17</td>
<td>42.84</td>
</tr>
<tr>
<td>cam_1_16-03-26_03_00_00.mkv_seq02</td>
<td>67.61</td>
<td>0.22</td>
<td>2.84</td>
</tr>
<tr>
<td>cam_1_16-03-26_03_00_00.mkv_seq03</td>
<td>0.00</td>
<td>0.00</td>
<td>0.04</td>
</tr>
<tr>
<td>cam_1_16-03-26_03_00_00.mkv_seq04</td>
<td>21.26</td>
<td>0.41</td>
<td>3.24</td>
</tr>
<tr>
<td>cam_1_16-03-26_03_00_00.mkv_seq05</td>
<td>21.26</td>
<td>0.25</td>
<td>72.64</td>
</tr>
<tr>
<td>cam_1_16-03-26_03_00_00.mkv_seq06</td>
<td>2.12</td>
<td>0.20</td>
<td>45.24</td>
</tr>
<tr>
<td>cam_1_16-03-26_03_00_00.mkv_seq07</td>
<td>40.93</td>
<td>0.23</td>
<td>56.68</td>
</tr>
<tr>
<td>cam_1_16-03-26_03_00_00.mkv_seq08</td>
<td>76.14</td>
<td>0.12</td>
<td>28.84</td>
</tr>
<tr>
<td>cam_1_16-03-26_03_00_00.mkv_seq09</td>
<td>23.72</td>
<td>0.65</td>
<td>49.24</td>
</tr>
<tr>
<td>cam_1_16-03-26_04_00_00.mkv_seq01</td>
<td>27.87</td>
<td>0.19</td>
<td>62.44</td>
</tr>
<tr>
<td>cam_1_16-03-26_04_00_00.mkv_seq02</td>
<td>26.76</td>
<td>0.17</td>
<td>47.84</td>
</tr>
<tr>
<td>cam_1_16-03-26_04_00_00.mkv_seq03</td>
<td>3.54</td>
<td>0.32</td>
<td>71.12</td>
</tr>
<tr>
<td>cam_1_16-03-26_05_00_00.mkv_seq01</td>
<td>22.49</td>
<td>0.13</td>
<td>67.24</td>
</tr>
<tr>
<td>cam_1_16-03-26_05_00_00.mkv_seq02</td>
<td>31.37</td>
<td>1.91</td>
<td>2.04</td>
</tr>
<tr>
<td>cam_1_16-03-26_05_00_00.mkv_seq03</td>
<td>9.86</td>
<td>0.31</td>
<td>14.2</td>
</tr>
<tr>
<td>cam_1_16-03-26_05_00_00.mkv_seq04</td>
<td>6.00</td>
<td>0.14</td>
<td>97.36</td>
</tr>
<tr>
<td>cam_1_16-03-26_05_00_00.mkv_seq05</td>
<td>34.36</td>
<td>0.26</td>
<td>84.04</td>
</tr>
<tr>
<td>cam_1_16-03-26_06_00_00.mkv_seq01</td>
<td>32.76</td>
<td>0.33</td>
<td>41.76</td>
</tr>
<tr>
<td>cam_1_16-03-26_06_00_00.mkv_seq02</td>
<td>9.56</td>
<td>0.11</td>
<td>19.24</td>
</tr>
<tr>
<td>cam_1_16-03-26_06_00_00.mkv_seq03</td>
<td>6.85</td>
<td>0.27</td>
<td>63.08</td>
</tr>
<tr>
<td>cam_1_16-03-26_06_00_00.mkv_seq04</td>
<td>4.52</td>
<td>0.25</td>
<td>46.04</td>
</tr>
<tr>
<td>cam_1_16-03-26_06_00_00.mkv_seq05</td>
<td>38.24</td>
<td>0.12</td>
<td>23.12</td>
</tr>
<tr>
<td>cam_1_16-03-26_06_00_00.mkv_seq06</td>
<td>12.18</td>
<td>0.11</td>
<td>29.24</td>
</tr>
<tr>
<td>cam_1_16-03-26_06_00_00.mkv_seq07</td>
<td>18.58</td>
<td>0.12</td>
<td>28.64</td>
</tr>
<tr>
<td>cam_1_16-03-26_06_00_00.mkv_seq08</td>
<td>15.05</td>
<td>0.15</td>
<td>64.04</td>
</tr>
<tr>
<td>Filename</td>
<td>Value 1</td>
<td>Value 2</td>
<td>Value 3</td>
</tr>
<tr>
<td>--------------------------------------------------------------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>cam_1_16-03-26_06_00_00.mkv_seq09</td>
<td>19.86</td>
<td>0.20</td>
<td>39.08</td>
</tr>
<tr>
<td>cam_1_16-03-26_06_00_00.mkv_seq10</td>
<td>10.51</td>
<td>0.20</td>
<td>30.44</td>
</tr>
<tr>
<td>cam_1_16-04-25_23_00_00.mkv_seq01</td>
<td>26.35</td>
<td>0.21</td>
<td>13.36</td>
</tr>
<tr>
<td>cam_1_16-04-25_23_00_00.mkv_seq02</td>
<td>9.37</td>
<td>0.21</td>
<td>70.04</td>
</tr>
<tr>
<td>cam_1_16-04-25_23_00_00.mkv_seq03</td>
<td>9.54</td>
<td>0.75</td>
<td>50.76</td>
</tr>
<tr>
<td>cam_1_16-04-26_00_00_00.mkv_seq01</td>
<td>15.59</td>
<td>0.48</td>
<td>76.44</td>
</tr>
<tr>
<td>cam_1_16-04-26_00_00_00.mkv_seq02</td>
<td>74.92</td>
<td>0.32</td>
<td>48.64</td>
</tr>
<tr>
<td>cam_1_16-04-26_00_00_00.mkv_seq03</td>
<td>41.93</td>
<td>0.44</td>
<td>48.84</td>
</tr>
<tr>
<td>cam_1_16-04-26_01_00_00.mkv_seq01</td>
<td>48.86</td>
<td>0.25</td>
<td>40.52</td>
</tr>
<tr>
<td>cam_1_16-04-26_02_00_00.mkv_seq01</td>
<td>15.23</td>
<td>0.23</td>
<td>132.6</td>
</tr>
<tr>
<td>cam_1_16-04-26_02_00_00.mkv_seq02</td>
<td>13.49</td>
<td>0.29</td>
<td>58.12</td>
</tr>
<tr>
<td>cam_1_16-04-26_03_00_00.mkv_seq01</td>
<td>12.55</td>
<td>0.23</td>
<td>64.04</td>
</tr>
<tr>
<td>cam_1_16-04-26_03_00_00.mkv_seq02</td>
<td>51.94</td>
<td>0.68</td>
<td>46.44</td>
</tr>
<tr>
<td>cam_1_16-04-26_04_00_00.mkv_seq01</td>
<td>21.54</td>
<td>0.18</td>
<td>7.8</td>
</tr>
<tr>
<td>cam_1_16-04-26_04_00_00.mkv_seq02</td>
<td>45.54</td>
<td>0.20</td>
<td>60.6</td>
</tr>
<tr>
<td>cam_1_16-04-26_05_00_00.mkv_seq01</td>
<td>37.14</td>
<td>0.52</td>
<td>23.48</td>
</tr>
<tr>
<td>cam_1_16-04-26_05_00_00.mkv_seq02</td>
<td>85.81</td>
<td>0.22</td>
<td>18.04</td>
</tr>
<tr>
<td>cam_1_16-04-26_05_00_00.mkv_seq03</td>
<td>25.37</td>
<td>0.40</td>
<td>40.84</td>
</tr>
<tr>
<td>cam_1_16-04-26_05_00_00.mkv_seq04</td>
<td>5.27</td>
<td>0.53</td>
<td>71.32</td>
</tr>
<tr>
<td>cam_1_16-04-26_05_00_00.mkv_seq05</td>
<td>14.01</td>
<td>0.09</td>
<td>36.84</td>
</tr>
<tr>
<td>cam_1_16-04-26_05_00_00.mkv_seq06</td>
<td>11.83</td>
<td>0.17</td>
<td>30.76</td>
</tr>
<tr>
<td>cam_1_16-04-26_05_00_00.mkv_seq07</td>
<td>14.79</td>
<td>0.13</td>
<td>35.16</td>
</tr>
<tr>
<td>cam_1_16-04-26_05_00_00.mkv_seq08</td>
<td>20.78</td>
<td>0.40</td>
<td>48.32</td>
</tr>
<tr>
<td>cam_1_16-04-26_05_00_00.mkv_seq09</td>
<td>13.87</td>
<td>0.16</td>
<td>28.84</td>
</tr>
</tbody>
</table>

Table C.1
## D. Module Test Tracking

<table>
<thead>
<tr>
<th>Video name</th>
<th>MOTP [cm]</th>
<th>Miss-rate [%]</th>
<th>FP [%]</th>
<th>Mismatches [#]</th>
<th>MOTA [%]</th>
<th>Persons</th>
<th>Trackers</th>
</tr>
</thead>
<tbody>
<tr>
<td>cam_1_16-03-21_23_00_00</td>
<td>61</td>
<td>45.0</td>
<td>39.0</td>
<td>20</td>
<td>15.5</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>cam_1_16-03-22_00_00_00</td>
<td>41</td>
<td>18.7</td>
<td>2.9</td>
<td>0</td>
<td>78.4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>cam_1_16-03-22_01_00_00</td>
<td>73</td>
<td>51.1</td>
<td>46.4</td>
<td>4</td>
<td>2.1</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>cam_1_16-03-22_02_00_00</td>
<td>78</td>
<td>92.5</td>
<td>89.2</td>
<td>0</td>
<td>-81.7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>cam_1_16-03-22_03_00_00</td>
<td>83</td>
<td>69.7</td>
<td>65.6</td>
<td>0</td>
<td>-38.3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>cam_1_16-03-22_04_00_00</td>
<td>68</td>
<td>67.3</td>
<td>24.1</td>
<td>0</td>
<td>8.5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>cam_1_16-03-22_05_00_00</td>
<td>60</td>
<td>39.8</td>
<td>37.1</td>
<td>0</td>
<td>23.1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>cam_1_16-03-22_06_00_00</td>
<td>47</td>
<td>28.8</td>
<td>28.8</td>
<td>2</td>
<td>42.3</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>cam_1_16-03-23_23_00_00</td>
<td>58</td>
<td>44.9</td>
<td>31.9</td>
<td>14</td>
<td>23.1</td>
<td>29</td>
<td>33</td>
</tr>
<tr>
<td>cam_1_16-03-24_01_00_00</td>
<td>53</td>
<td>67.7</td>
<td>18.9</td>
<td>16</td>
<td>13.3</td>
<td>17</td>
<td>21</td>
</tr>
<tr>
<td>cam_1_16-03-24_02_00_00</td>
<td>44</td>
<td>39.0</td>
<td>11.7</td>
<td>0</td>
<td>49.3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>cam_1_16-03-24_03_00_00</td>
<td>40</td>
<td>43.1</td>
<td>29.2</td>
<td>5</td>
<td>27.6</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>cam_1_16-03-24_04_00_00</td>
<td>54</td>
<td>79.1</td>
<td>12.8</td>
<td>2</td>
<td>8.1</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>cam_1_16-03-24_05_00_00</td>
<td>52</td>
<td>45.1</td>
<td>34.3</td>
<td>5</td>
<td>17.3</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>cam_1_16-03-24_06_00_00</td>
<td>47</td>
<td>24.9</td>
<td>19.4</td>
<td>2</td>
<td>55.7</td>
<td>26</td>
<td>28</td>
</tr>
<tr>
<td>cam_1_16-03-25_23_00_00</td>
<td>47</td>
<td>64.8</td>
<td>26.8</td>
<td>47</td>
<td>8.3</td>
<td>43</td>
<td>53</td>
</tr>
<tr>
<td>cam_1_16-03-26_00_00_00</td>
<td>47</td>
<td>58.3</td>
<td>15.1</td>
<td>10</td>
<td>26.4</td>
<td>16</td>
<td>14</td>
</tr>
<tr>
<td>cam_1_16-03-26_01_00_00</td>
<td>42</td>
<td>63.3</td>
<td>12.5</td>
<td>6</td>
<td>24.1</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td>cam_1_16-03-26_02_00_00</td>
<td>49</td>
<td>36.3</td>
<td>20.2</td>
<td>7</td>
<td>43.2</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>cam_1_16-03-26_03_00_00</td>
<td>48</td>
<td>54.8</td>
<td>22.9</td>
<td>1</td>
<td>22.3</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>cam_1_16-03-26_04_00_00</td>
<td>46</td>
<td>63.7</td>
<td>23.5</td>
<td>12</td>
<td>10.3</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>cam_1_16-03-26_05_00_00</td>
<td>35</td>
<td>31.0</td>
<td>12.6</td>
<td>0</td>
<td>36.4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>cam_1_16-03-26_06_00_00</td>
<td>47</td>
<td>27.9</td>
<td>20.7</td>
<td>2</td>
<td>51.2</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>cam_1_16-04-25_23_00_00</td>
<td>46</td>
<td>30.1</td>
<td>22.7</td>
<td>3</td>
<td>46.9</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>cam_1_16-04-26_00_00_00</td>
<td>51</td>
<td>54.4</td>
<td>31.7</td>
<td>5</td>
<td>13.7</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>cam_1_16-04-26_01_00_00</td>
<td>61</td>
<td>21.8</td>
<td>8.0</td>
<td>0</td>
<td>70.1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>cam_1_16-04-26_02_00_00</td>
<td>64</td>
<td>53.7</td>
<td>50.1</td>
<td>3</td>
<td>-4.0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>cam_1_16-04-26_03_00_00</td>
<td>47</td>
<td>35.8</td>
<td>0.0</td>
<td>0</td>
<td>64.2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>cam_1_16-04-26_04_00_00</td>
<td>62</td>
<td>80.3</td>
<td>80.3</td>
<td>0</td>
<td>-60.7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>cam_1_16-04-26_05_00_00</td>
<td>41</td>
<td>19.5</td>
<td>13.0</td>
<td>0</td>
<td>67.5</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

| ALL                      | 49        | 53.2          | 24.4   | 166            | 22.3     | 295     | 325      |

**Table D.1:** Results from test data by comparing annotated positions in each frame to the tracker’s positions. The miss-rate is the number of unassigned positions divided by the total number of positions. The FPs are trackers created due to noise (assumed to be persons) over the total number of trackers. The mismatches describe the number of times trackers change persons. The MOTA is the trackers accuracy when tracking, irregardless of persons. The MOTP is the tracker’s deviation from ground truth.
E. Tracking Training Output