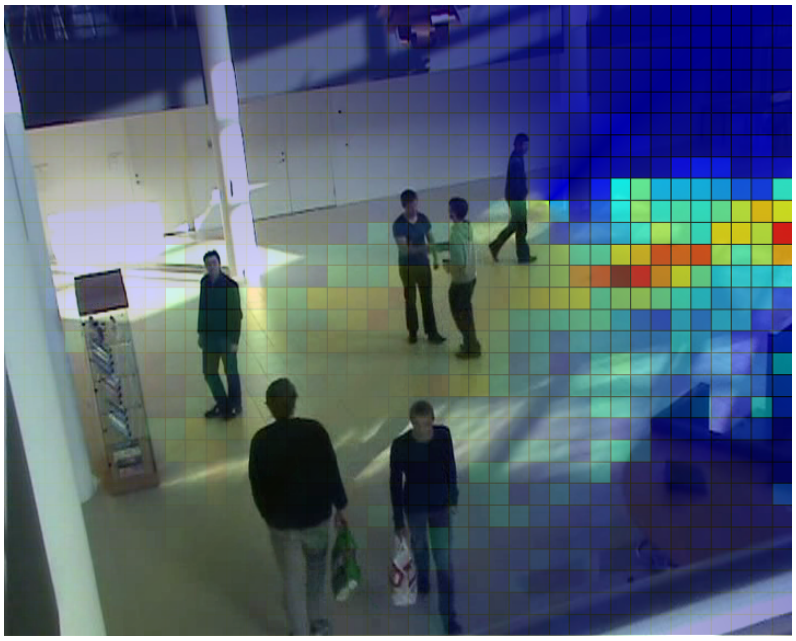
Automatic Estimation of Statistics on the Movement of People in Complex Indoor Scenes



Group 1020

Thomas Ægidiussen Jensen
Henrik Anker Rasmussen

Master's thesis in Vision, Graphics and Interactive Systems
Aalborg University, autumn 2010 - spring 2011

AALBORG UNIVERSITET

The Faculties of Engineering, Science and Medicine
Department of Electronic Systems
Fredrik Bajers Vej 7
Phone: +45 96 35 86 00
http://es.aau.dk

Title:
Automatic Estimation of Statistics on the Movement of People in Complex Indoor Scenes.

Project period:
September 2, 2010 to May 31, 2011

Project group: 11gr1020

Group members:
Thomas Ægidiussen Jensen
Henrik Anker Rasmussen

Supervisor:
Thomas B. Moeslund

Number of copies: 4

Number of report pages: 105

Total number of pages: 133

Appended documents:
5 appendices, 1 CD

Finished: May 2011

Abstract:

This project deals with the problem of automatically generating statistics on the movement of people in complex indoor scenes. These statistics are to be used for commercial purposes, e.g. to determine rental prices of shops, placement of ads, etc. Commercial solutions for this exist today, ranging from simple solutions like infrared beams to advanced and expensive solutions like thermal sensors. However, common for these are that they require extra hardware to be installed. Furthermore, existing solutions that are able to generate advanced statistics on the movement of people require expensive hardware.

In this project, a system is developed that automatically generates statistics on the movement of people using footage from surveillance cameras in the Friis shopping mall. Thus, the system developed in this project requires no installation of extra sensor hardware. Furthermore, it requires no prior knowledge about the scene.

The system consists mainly of a multi-human tracker and a visualizer. Prior to generating the actual statistics, the system estimates information about the scene on a long video sequence to improve the performance of the multi-human tracker.

The system is evaluated quantitavely and qualitatively on video from the Friis shopping mall and on video recorded for this work at Aalborg University. The results show that while a higher error is measured in the statistics generated on video from the Friis shopping mall, the generated statistics generally resemble the actual statistics. Thus, it is seen that the system is able to capture the trends of the movement of people and make a visual representation of these.

This project makes the following contribution to the problem of tracking multiple persons in complex scenes: Enhancing the output from human detectors by exploiting scene information automatically generated from reliable human detections. Also, estimated trajectories are used for generating statistics on the movement of people. Lastly, suggestions on possible improvements and extensions of the system are given.

Preface

This report constitutes the master's thesis for Thomas Ægidiussen Jensen and Henrik Anker Rasmussen at the Vision, Graphics and Interactive Systems specialization at Department of Electronic Systems, Aalborg University. The project work spanned the period of September 2, 2010 to May 31, 2011.

The report describes the development of a system that generates statistics on the movement of people inside the Friis Shopping Mall. The report consists of an analysis, where the problem of generating statistics on the movement of people is analyzed, which leads to a Requirement Specification and an Acceptance Test Specification for the system. This is followed by chapters that describe the structure of the system and the design of its underlying subsystems and modules. The system is evaluated through an Acceptance Test, which leads to a conclusion on the project, followed by suggestions on future work that can be done to improve the system. This is followed by appendices used to describe theory used to support the description of the design of the modules. Lastly, the sources used throughout the report are listed in the Bibliography chapter.

The authors would like to thank Katrine Velter, Mille Toft Schou, Susanne Paulsen and Søren Hilker for their valuable assistance and for providing surveillance footage from the Friis Shopping Mall for use in this project.

Reading Instructions

Some words and expressions that are used throughout the report, are written correctly the first time they are used, followed by an abbrevation stated in parantheses. This abbrevation is used hereafter. For instance, Histogram of Oriented Gradients (HOG) will be written as HOG after the first occurence. An overview of the abbreviations and notations used throughout the report is provided in the following chapter, "Nomenclature".

References are written as numbers in square brackets, e.g. [1]. This number then corresponds to the number written in the Bibliography chapter, where information about the source, such as the name of the author, the name of publication and the publisher, etc. can then be found. The Internet sources are included on the attached CD in the back of this report.

CD

The CD attached to this report contains the used internet sources (as PDFs), C++ source code files, Matlab source code files, and samples of the video data used for developing and evaluating the system. The contents are arranged in folders as described below:

- `bibliography/`
  - This folder contains the internet sources as PDFs.
- `sources/`
  - This folder contains the source code for the system developed in this work.
- `videos/`
  - This folder contains videos with estimated trajectories.
- `report.pdf`
  - This report in PDF format.

Aalborg University, May 31, 2011.

Thomas Ægidiussen Jensen	Henrik Anker Rasmussen

Nomenclature

In this chapter, the abbreviations and the mathematical notations that are used throughout this report are listed.

Abbreviations

The abbreviations used throughout this report are listed in Table 1.

Abbreviation	Phrase
AdaBoost	Adaptive Boosting.
CCTV	Closed-Circuit Television.
FOV	Field Of View.
FP/FN	False Positive/False Negative.
FPS	Frames Per Second.
Friis	The Friis shopping mall.
GT	Ground Truth.
HOG	Histogram of Oriented Gradients.
HSV	Hue, Saturation, Value.
HMM	Hidden Markov Model.
JPDAF	Joint Probability Data Association Filtering.
k-NN	k-Nearest Neighbor.
LBP	Local Binary Patterns.
MHT	Multiple Hypothesis Tracking.
MIL	Multi Instance Learning.
MOTA	Multi-Object Tracking Accuracy.
MOTP	Multi-Object Tracking Precision.
PCA	Principal Component Analysis.
pdf	Probability Density Function.
PMHT	Probabilistic Multiple Hypothesis Tracking.
RGB	Red, Green, Blue.
RGI	Red, Green, Intensity.
RMSE	Root Mean Squared Error.
SIFT	Scale-Invariant Feature Transform.
SMAPE	Symmetric Mean Absolute Percentage Error.
SURF	Speeded Up Robust Features.
SAD	Sum of Absolute Differences.
SVM	Support Vector Machine.
TP	True Positive.

Table 1: *Abbreviations used in this report.*

Notations

The mathematical notations used in this report are listed in Table 2.

Notation	Description
$\mathbf{A}$	Denotes a matrix.
A_{ij}	Denotes element ij in $\mathbf{A}$.
$\mathbf{a}$	Denotes a vector.
a_i	Denotes element i in $\mathbf{a}$.
c	Denotes a scalar.
$\lvert\cdot\rvert$	Denotes an absolute value.
$\lvert\lvert\cdot\rvert\rvert$	Denotes the L_2 norm.
$\lvert\lvert\cdot\rvert\rvert_p$	Denotes the p-norm.

Table 2: *Mathematical notations used in this report.*

Table of Contents

Chapter 1

Introduction

Analyzing the movement of people is a relatively young field of research. In the recent years, advances in tracking technology has made analysing the movement of entities reasonable. [2] Applications of analysing the movement of entities have a wide range, spanning from analyzing the traffic to the movement of animals, the movement of people, etc.

Figure 1.1 shows an example of the movement of four entities in a number of time steps. In this example, the objective might be to detect certain movement patterns of the entities. Two movement patterns are shown; a flock with three entities and a meeting place.

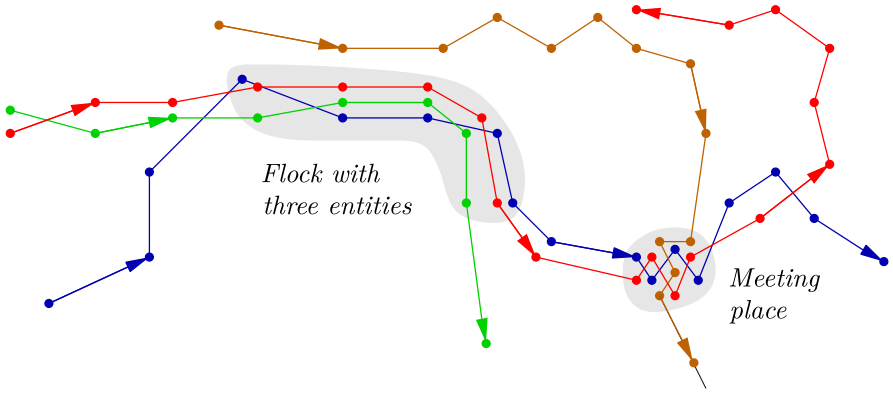


Figure 1.1: *An example of the movement of four entities. Image courtesy of [2]*

The movement of people has been analyzed in previous works, e.g. in [3], where the movement behavior of people in a park in Denmark was analyzed. Another application is to analyze the movement of people in indoor environments. For commercial purposes, such analyses can potentially be a valuable tool e.g. for shopping malls like the Friis shopping mall.

Friis is a shopping mall in Aalborg that opened in the spring 2010. It houses shops, cafes and restaurants, among other things. Statistics on the movement of people in the shopping mall can be used by the managers of the shopping mall to determine the rental prices of shops, placement of ads, etc.

This project deals with the problem of estimating statistics on the movemnt of people in indoor environments for commercial purposes. The goal is to develop a system that can estimate reliable statistics that capture trends in the movement of people in the Friis shopping mall that are of relevance for the managers of Friis. Furthermore, the goal is to develop such a system, where the costs in terms of hardware requirements are kept at a low.

1.1 Initial problem

The introduction leads to the following initial problem:

How can a system that is able to estimate statistics on the movement of people in complex indoor scenes for commercial purposes be developed?

This question is answered in the next chapter through an analysis of the problem.

Chapter 2

Analysis

Before setting up the requirements of the system, the use of surveillance in general is analyzed. First, the general use of surveillance today is investigated; how much is it used today and why, and what ethical issues might there be in using surveillance. Next, available technologies for estimating statistics on the movement of people are described and evaluated in terms of what technology that is the most suited for this work. After this, different abstraction levels are investigated to determine, which level that is realizable for this work. When the abstraction level has been determined, the movement statistics that are to be estimated by the system developed in this work are defined. This is followed by an investigation of the related work in the area, i.e. an overview of the previous work is given together with a description of the current state of the art in person tracking using the chosen technology. A number of approaches are then compared in order to find the most useful approach for this work. It ends up with a problem description defining the limitations of the systems and a problem statement.

2.1 Surveillance today and in general

In this section, the use of surveillance is examined. First why surveillance is used. Then, how much surveillance is used today and lastly, ethical issues in using surveillance are investigated.

2.1.1 Why use surveillance

The topic of monitoring people is a well-debated topic. One of the reason for this is that newer technologies make acquiring and storing information about people easier and cheaper for the authorities. Social networks even encourage people to share information about themselves and thereby making it even easier for companies or authorities to monitor people.

One of the applications for monitoring people is the use of video surveillance, which has become a normal part of most people's daily life. Whenever a person goes to the post office, shopping mall or travel with bus, he or she is most likely to be monitored by video cameras. The cameras are connected in systems, where all the images recorded are stored for later use. The systems are becoming increasingly intelligent by being able to detect motion and thereby only record videos of interest. [4] Most Danes are happy with the video surveillance in Denmark today. According to a survey from 2009, 85% of the Danish people disagree with the claim that there is too much video surveillance in Denmark and 82% think that surveillance provides them security.[1]

The following sections describe notable uses of surveillance.

Crime prevention

Often when arguing in favor of using video surveillance, it is said that the use of video surveillance prevents crimes like bank robberies, shoplifting and assaults. The cameras are said to have a

preventive effect on otherwise potentially criminal persons. Other arguments for using surveillance is that it often makes crime-solving easier for the police after the event has occurred, as they might be provided with identities of the persons committing the crimes.[5, p. 1]

The Danish government used Jomfru Ane Gade in Aalborg as a test case for how video surveillance affects the behavior of people in a street with mainly restaurants, bars and night clubs. In this street, violence and disturbance occurs frequently. The test case started at July 11, 2008 and was evaluated by the police at the end of 2008. It was concluded that the surveillance caused a slight reduction in the number of reported incidents in the period. However, this cannot be guaranteed to be a result of the surveillance. The videos from the cameras were used in 10 cases of violence, where they were primarily used for identification of the criminals. The police also discovered an increase of 10% in solving cases that occurred in Jomfru Ane Gade.[6]

Making citizens feel secure

Another argument for using surveillance is that it makes the citizens feel more secure. According to a survey from 2009, surveillance makes 82% of the Danish people feel more secure.[1] In the evaluation of the Jomfru Ane Gade test case, a survey was conducted on what the staff working in the street thought about the video surveillance. It showed that the staff felt more secure when working and moving around in the street.[6]

Personalized Marketing

Personalized marketing is when companies differ their advertisements from person to person. This kind of marketing is widely used on the Internet, where web pages like the online search engine Google monitors each user's searches and previously visited web pages to learn what the user is interested in. This is then used to make advertisements directed at the user's interests. [7]

The same concept could in the future be transferred to real-life advertisement signs. If computers were capable of analyzing video surveillance data and thereby were capable of identifying groups of people interested in certain kinds of products, they could update an advertisement screen with commercials aimed specifically at the person looking at the screen.

Marketing Analysis

When shop managers decide where to place products in their stores, it is not done randomly.

A lot of efforts are done to optimize the shopping environment to improve the customer's shopping experience and improve opportunities to show tempting products when the shopper is receivable for it. When marketers make analyses in this area, video surveillance is used to keep track of customers and their movement in the store. The inspection of the videos, which is done manually, can reveal the behavior of the costumers, such as which aisles they prefer to walk through, and where they spend most of their time. This can be used for strategic placement of products that need more attention, and to communicate to the customer. Information about where in the store crowd conditions like bottle necks occur is also useful for improving the customer's experience by enhancing the flow in the store. [8]

2.1.2 How much is surveillance used today

Today, cameras are widely used to monitor the public domain. In 2006, closed-circuit television (CCTV) systems in the streets of Britain used up to 4.2 million surveillance cameras to monitor citizens. In average, a Briton was watched by CCTV 300 times a day. [9] In Denmark, surveillance cameras are also widely used. In 2008, it was estimated that at least 250,000 cameras were used for monitoring citizens in Denmark. The tendency in the use of surveillance is clear; the use of surveillance is increasing rapidly. In 2004, there were about 100,000 surveillance cameras i Denmark, whereas in 2008 there were more than 250,000. [10] [11] At July 1, 2007 a new law concerning video surveillance was introduced, which allowed for more surveillance in public. This law appears to be well used since the private and the public sector are increasing the number of surveillance cameras. In 2007, it was estimated that more than 25,000 cameras are installed

every year in Denmark. The Danish state devoted 10 million DKK in 2008 for installation of video surveillance in criminally affected areas. [5, p. 1]

2.1.3 Ethical issues

Public monitoring of people has often led to concerns among the population. People have long had a fear of the surveillance being abused by the authorities. The insecurity of being publicly surveilled goes back several decades. An early example of this is the book "Nineteen Eighty-Four" by George Orwell, which was released in the beginning years of the cold war. The book described a dystopian world in the year of 1984, where everyone were under video surveillance by the authorities. Citizens were constantly reminded of this through the phrase "Big Brother is watching you". To this day, the term "Big brother" is still used as a synonym for abuse by the authorities. [12]

As the technology for video surveillance has matured and cheapened, the extent of video surveillance has increased similarly. The increasing extent of surveillance is also caused by the attempts of ensuring the national securities by the governments, as a result of concerns for terrorism, immigration and border control. [13] Along with this progress, the public debate regarding the ethical issues concerning surveillance has achieved much attention in the recent years. This debate is, to a great extent, focused on the question of whether surveillance helps reducing the number of criminal activities or not. The debate divides the population into two camps that can be described as "surveillance optimists" and "surveillance pessimists", where the former argue that "if you have not done anything wrong, then you have nothing to fear". The "surveillance pessimists" argue that "Big Brother sees everything, privacy is dead". The claims in the public debate are often contradictory. Some claim that surveillance reduces crime, and thus ensuring the safety of the citizens. Others claim that surveillance does nothing but move the crimes to other places, and thereby providing a false feeling of security for the citizens. [5]

The attitude among the Danish population towards surveillance has changed over the years as a result of the increased terror threat. Furthermore, the younger generation has an entirely different view on surveillance than the elderly that remembers the cold war with the espionage and wire tapping. Most Danes think that as they are not criminals, surveillance does not bother them. This is supported by a study, which shows that one out of 20 is bothered by surveillance in the daily life. [14]

2.2 Commercial products

In this section, a number of available products and technologies for generating statistics on the movement of people in indoor environments are described. This is to give insight in what is possible in the field today, and to give an overview of what advantages and disadvantages, the existing technologies have compared to Friis.

2.2.1 Tally counters

A tally counter is a mechanical or digital counting device, which has a button to increase the count by one. In Figure 2.1a a tally counter is shown. The counter is operated by a person watching an area, where the number of people moving through is desired. The operator is pushing the button every time he or she sees a person passing through the area. The method is flexible, since it does not require any physical configuration, other than the tally counter operator having to stand, where he or she can see the area. The results can be accurate, since the intelligence of the person is used for determining, when a human is passing through the area. Using a tally counter is a manual job requiring a person dedicated for this. Also, the method provides simple results not adequate for generating advanced statistics. [15]

2.2.2 Infrared beams

This technology consists of a transmitter that sends out a beam of infrared light and a receiver that counts every time the beam is interrupted. The transmitter and receiver are placed on each

(a) (b)

Figure 2.1: *Commercial products for people counting. (a): Tally counter (Courtesy http://www.armynnavy.com) (b): Thermal cameras located in top of the image in the Friis shopping mall.*

side of an entrance to the area in question in a height that fits the average chest-height of the persons passing through. The count can either be displayed on an LCD display on the receiver or can be transmitted to a computer. If two infrared beams are placed close to each other, they can determine the direction of a person passing through, so that the system knows if the person is going in or out of the area. [16]

This method of detecting people has the advantage that it is easy to set up. The only configuration is to set up the transmitter and receiver in an entrance to the area. The method has problems with detecting people in places, where they can walk beside each other so that they are not separated when passing the beam(s). If a passing person decides to stand in the entrance, so that the beam is blocked, it will not detect other persons passing through. Another problem with this technology is that it can miss people passing through if the receiver is in direct sunlight. For using infrared beams, they will have to be set up in Friis. For generating advanced statistics, a large number of infrared beams would have to be installed, although the output would still lack temporal information.

2.2.3 Thermal sensors

Thermal sensors are mounted on top of an entrance and are able to register multiple persons passing through simultaneously. In Figure 2.1b the thermal sensors used by Friis. They pick up the heat from humans with the thermal sensor array. Embedded software is analyzing the data to find high concentrations of thermal infrared radiations within the field of view (FOV) of the array. This technology does not depend on static scenery or foreground-background segmentation, since the camera only senses heat. This makes the technology invariant to changes in illumination. Tracking people with this method provides a high accuracy (approximately 98%).[17] A thermal sensor with a 60° FOV might have to be mounted in a height of 2.5m to 3.5m, meaning that the area covered has a width of approximately 3m to 5m. [18] [19] Use of this technology requires that thermal sensors must be set up in Friis. Furthermore, to cover large areas, many thermal sensors have to be installed.

2.2.4 Computer vision

The technology of tracking people in images from normal cameras is not wide spread in commercial use yet. This might be caused by the heavy computations required to track humans in images from complex scenes. Furthermore, the technology faces challenges, when the camera is placed so that occlusion of people by objects or other people can occur and when the camera is placed at a location with significant variations in the light. However, a lot of research has been done in

this area to solve these problems. This is investigated in the "Related Work" section (Section 2.5). Using computer vision requires no installation of new hardware, as the inside of Friis is almost covered by surveillance cameras as described in the following subsection.

2.2.5 Comparison

Table 2.1 shows an overview of the commercial products and technologies described. The table lists their advantages and disadvantages compared to Friis.

Product/technology	Advantage(s)	Disadvantage(s)
Tally counters.	Accurate.	Operated manually. Inadequate output.
Infrared beams.	Easy to set up.	Inaccurate (occlusion). Requires new hardware. Sensitive to sunlight.
Thermal sensors.	Accurate.	Requires new hardware.
Computer vision.	Accurate. New hardware not required.	New technology[1]; might face challenges.

Table 2.1: *Overview of commercial products and technologies showing their advantages and disadvantages compared to Friis.*

The managers in Friis are using thermal cameras to keep track of how many people that are inside the mall at a given moment. At every entrance of mall, thermal cameras that count the number of people entering and leaving are installed. This means that the managers in Friis only knows how many people there are inside the mall at a given time, and not where inside of the mall they are.

If the challenges with the computer vision technology for tracking people can be overcome, a great amount of money might be saved compared to other possible solutions and still it will be possible to generate advanced statistics on people's movement.

The inside of Friis almost covered entirely by surveillance cameras used for crime prevention. For this work, it is chosen to use these surveillance cameras together with computer vision to generate statistics on the movement of people inside Friis.

2.3 Abstraction Levels

When dealing with computer vision, different levels of abstraction of the output information can be used, depending on both what level of detail that is necessary for the system and also on what is possible within the specific scenario. The higher the level, more detailed information is obtained. Similarly, a higher abstraction level would usually result in a more difficult and longer development process. In this section, three abstraction levels are defined and it is determined, which abstraction level that is realizable for this work.

The highest level of abstraction is where a system is able to track humans in an image and determine what they are doing. This covers both what they are looking at and what kind of activity they are doing, which might include waiting or stealing. A system, which is able to do this could be used for crime prevention by starting an alarm, e.g. when it detects a person stealing. It could also be used for market analysis. This could be analyzing where it is convenient to set up advertisement, since it can tell where people dwell or where people tend to look. Such a system would be advanced to develop, but would also be useful for marketing, as discussed in Section 2.1.1.

[1] "New" in this context means new in the area of surveillance relative to the listed products and technologies.

The middle level is when a system is able to determine, where in an image individual persons are found and then is able to track these persons. Such a system could be used for estimating the trajectories of people.

The lowest abstraction level is when a system is able to detect if something is in front of the camera, without being able to determine what it is.

As described in the Introduction, this work focuses on generating statistics on the movement of the people for commercial purposes. Thus, for the highest abstraction level, it will be relevant to track humans and also estimate, in which direction they are looking. A simple way to do the latter is to estimate the head pose. However, according to a survey on head pose estimation from 2009, no system has yet met all crucial design criteria. These criteria include accuracy, monocular view, no need for initialization, full range of head motion and multiple persons in one image. Furthermore, most of the systems make assumptions that limit their applicability to real-world scenarios. For instance, many of these systems only apply to near-field images. [20]

The middle level of abstraction is tracking humans. It has been shown that it is indeed possible today to track multiple (possibly occluded) persons in complex scenes. Systems that are able to do this from a monocular view with a low error rate have been proposed. [21] [22] [23] Thus, the middle level is an obvious choice of abstraction level for this work. Another possibility is to choose the lowest abstraction level, i.e. a system that detects if something is in front of the camera, without taking into account what it is. However, instead of just detecting and counting the number of objects that are in front of the camera, detecting humans and tracking them over a period of time to generate statistics on their movement has the potential to yield better results. Therefore, to achieve the best results, it is chosen to develop a system that generates statitics on the movement of people by detecting and tracking them over time.

2.4 Movement Statistics

In this section, the problem of representing the movement of persons inside Friis statistically is analysed. The statistics are generated from trajectories estimated using computer vision. For this, previously used methods to represent the movement of people are investigated briefly. This is followed by a discussion on what information that is relevant to extract, i.e. what kind of information that is relevant for Friis. It ends up with a final list of stastitics that will be extracted and represented by the system.

Previously used methods for representing the movement of people statistically include calculating the optical flow, the person density and the mean speed. The optical flow is the movement of pixels in an image across frames and the person density is the number of persons inside an area.[24] [25] These statistics are however low level compared to the kind of statistics that Friis are interested in. The optical flow shows all movement in the image, whereas the person density and mean speed are simple numbers that do not hold much relevant information.

Instead of detecting all the movement in a scene by calculating the optical flow in the image, a compact, yet more relevant and expressive information for this work can be obtained by calculating the person flow. This is calculated by tracking persons in an image over a time period, while saving their trajectories. After this, the image is divided into a grid, where the average movement, in the form of various statistics is calculated for each field, yielding what is defined in this work as the person flow. The dimensions of the grid can be specified by the user. The principle of dividing the image into a grid and then extracting statistics independently for each field from person trajectories will be the base for extracting the movement statistics.

The statistics are to be used for commercial purposes, i.e. they are to be used for determining rental prices of shops, placement of ads, etc. Therefore, the system must be represent, where people are walking, and at what speed. Furthermore, the system must be able to show, where people tend to dwell. These statisics are represented as heat maps with dimensions equal to the grid dimensions. In this work, a heat map is defined as a histogram containing spatial information which has been color coded.

To extend the statistics, the directions of the persons within each field of the grid is represented as two vector plots, where the first vector plot shows the mean directions of each field, and the

second shows the four most frequently observed directions. Lastly, temporal information is shown by plotting a curve that shows the number of persons in the scene during the time period.

The following list comprises the statistics that are chosen to be extracted by the system, i.e. position, direction, speed, dwell analysis and the number of persons. Each type is followed by a description of how it is represented.

- Person Position
  - The number of persons walking through each bin. Represented as a heat map, see Figure 2.2.
- Speed
  - The average speeds of persons represented as a heat map, see Figure 2.2.
- Dwell Analysis
  - Heat map showing where persons tend to dwell. That is, a heat map illustrating numbers of people whose speed drop below a threshold value, see Figure 2.2.
- Direction
  - 2D plot showing a vector field of the mean directions, where all observed directions within the individual fields are summarized. The resulting direction vectors are plotted as shown in Figure 2.3a.
  - 2D plot showing the four most frequently observed directions. The four direction vectors within each field are color coded in red, green, cyan and blue, with red and blue being the most and least frequently observed directions, respectively. The length of the vectors illustrate the relationship between the observation frequency of the directions. This statistic is refered to as Top-Four from hereon. An example is shown in Figure 2.3b.
- Number of Persons
  - A curve showing the number of persons in the scene as a function of time.

Figure 2.2 shows an example of a heat map, where an image is divided into a grid, with each field making up a bin. The color of the bins express the value of the bins, e.g. the number of persons that have passed through. Figure 2.3 shows an example of an image divided into a grid showing the mean directions.

2.4.1 Intermediate Output

The ”Number of persons” statistic shows the number of persons during the time period. However, the ”Position”, ”Speed”, ”Dwell Analysis” and ”Direction” statistics are all accumulated and averaged over the entire time period. Thus, they do not contain any temporal information. To compensate for this, intermediate output can be generated within a number of equally sized intervals, specified by the user to complement the final statistics. This provides the user with an overview of the time trend of the statistics. The principle is illustrated in Figure 2.4 for one bin.

Automatic Detection of Behavior Change

Besides providing the user with an overview of the time trend of the statistics, the intermediate output can be used by the system to automatically detect behavior change. By making the system process a sequence spanning a large time period, e.g. a week, the system can estimate the typical intermediate plots. From this point, whenever an intermediate plot is estimated, the system can detect a behavior change by determining, whether the intermediate plot is significantly different from the typical intermediate plot generated in the same relative time interval. This can be useful, for instance for detecting when a change in the scene like newly put up advertisements affects the movement of people.

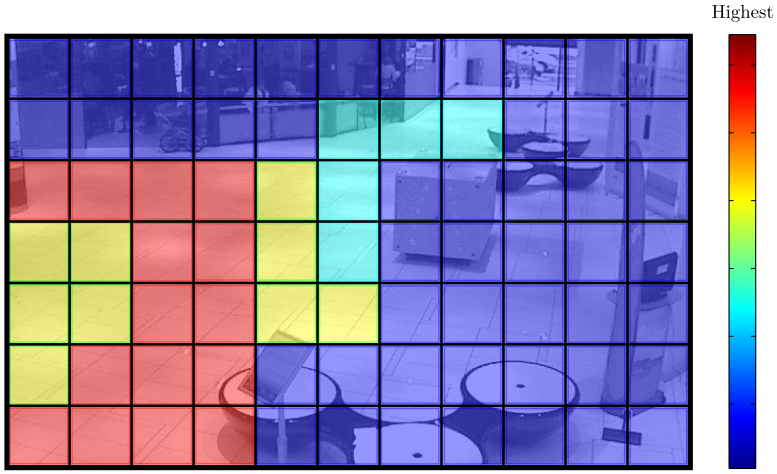
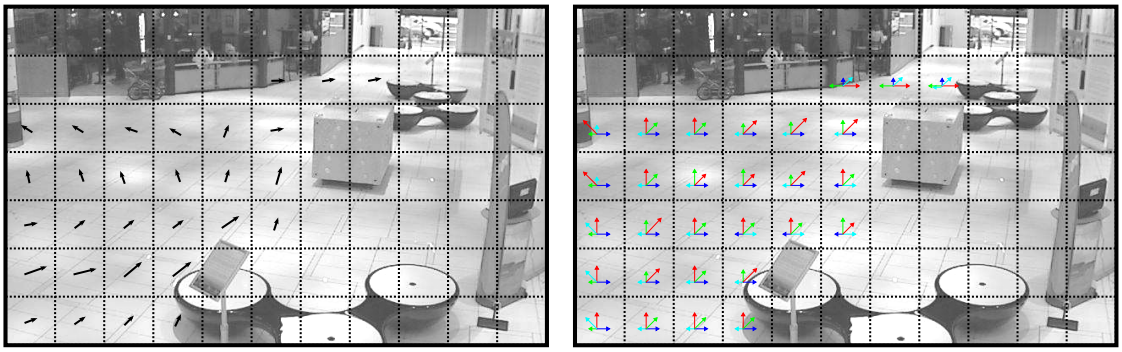


Figure 2.2: *An image divided into a grid, where each field make up a bin in a heat map drawn on the image. The image is a frame from a sequence recorded by a surveillance camera in Friis.*



(a) **(b)**

Figure 2.3: *An image divided into a grid. In Figure (a), each field contains its mean direction vector. In Figure (b), each field contains the Top-Four directions. The image is a frame from a sequence recorded by a surveillance camera in Friis.*

Thus, a final list of the information that is extracted by the system has been set up. This has been done after a brief investigation of previous methods and a discussion on what is relevant for this work with respect to Friis. This list will be the base of the system design and will be referred to the movement statistics from hereon.

2.5 Related Work

In this section the research within the area of tracking persons in video sequences is investigated. Tracking of single and multiple objects in general has received much attention by researchers. [21] Tracking persons is a special case of object tracking, which is of relevance for this work. Therefore, the related work within object tracking will be addressed with a focus on person tracking. First, the previous work in the area is examined. The approaches described in the previous work section will be divided into different categories. After this, an overview of the current state of the art will be given, where the state of the art approaches will be evaluated with respect to this work. This section is based on a survey on object tracking by A. Yilmaz, O. Javed and M. Shah [26], published in December 2006 and a number of articles published since 2006.

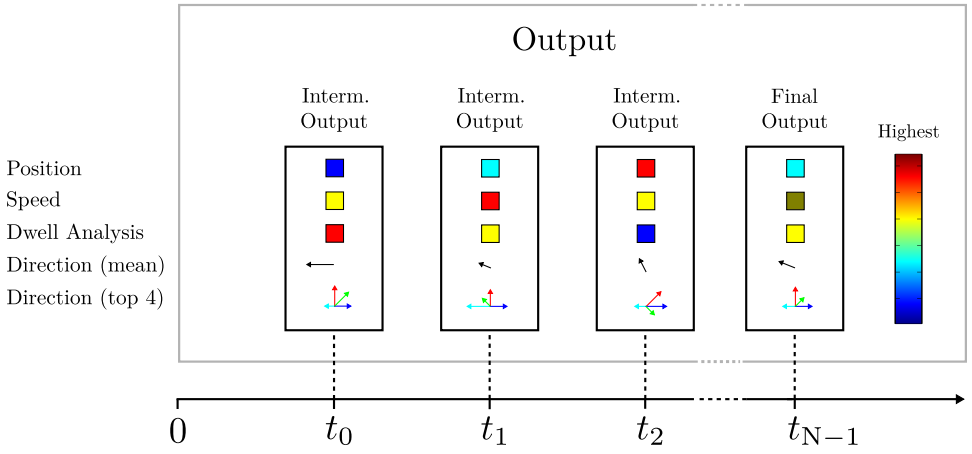


Figure 2.4: *Example of possible output from the system. For simplicity, one bin is shown. Intermediate heat maps and vector plots are generated within a specified number of equally sized intervals, N. Lastly, the final heat maps and vector plots over the entire time period are generated.*

2.5.1 Previous Work

This section describes the previous work within object tracking. It is split into the following subsections. The title of each subsection in the list below is followed by a brief description.

1. **Object representation**. This subsection investigates the different ways of representing an object visually that have been used.
2. **Feature selection**. The different features that have been used to describe objects are described here.
3. **Object detection**. Before an object can be tracked in an image, it must first be detected. Methods for this are looked into in this subsection.
4. **Object tracking**. This subsection describes the different methods that have been used to track objects in images.

Thus, the different ways of representing an object visually are described first.

Object representation

For an object to be tracked, it must also be represented. This has been done both on shape and appearance or a combination of both. When making representations of objects based on shape, basic methods have been used in research. These are listed below. Figure 2.5 shows the different types of object representations.

- **Points**. Points can represent an object by indicating the position of the object center, or by a number of points scattered on the object. The one point variant is often used when tracking small objects. See Figure 2.5a and 2.5b.
- **Simple shapes**. These could be ellipses or rectangles. They are used for rigid objects, but have also been used for humans[21]. See Figure 2.5c and 2.5d.
- **Articulated shape models**. Representing e.g. a human body as an articulated model means that each rigid body part is represented by simple shapes, like an ellipse or a rectangle. These body parts are then connected through joints. See Figure 2.5e.

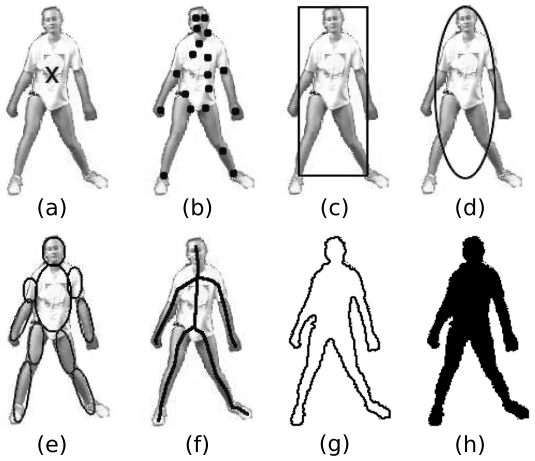


Figure 2.5: *Object representations. (a) and (b) show single and multiple point representations. (c) and (d) show simple shape representations; rectangle and ellipse. (e) shows an articulated shape model. (f) shows a skeleton model. (g) and (h) show a contour and a silhouette (image courtesy of [26]).*

- **Skeleton models**. A skeleton of an object can be generated by a medial axis transformation, which can reduce a silhouette of an object to a thin line. Both rigid and articulated objects can be represented using skeleton models. See Figure 2.5f.

- **Silhouette and contours**. A contour is the edge of an object, while a silhouette is the area inside the edge. These kinds of representations are used for non-rigid objects, and can reveal the pose of the non-rigid object.[27] See Figure 2.5g and 2.5h.

Objects are also represented by appearance, and the mostly used representations on appearance are listed here. Sometimes, the appearance is combined with a shape.

- **Probabilistic appearance models**, i.e. probability density estimates of the object appearance. The appearance of an object can be modeled inside the object's region, which is defined by simple shapes like rectangles or by shapes like silhouettes. The model can be parametric, such as a Gaussian or a mixture of Gaussians distribution of the colors or it can be non-parametric such as histograms of the colors inside the object's region. [21]

- **Templates and histograms**. Templates are made based on the object's region, defined by rectangles or silhouettes. The upside with this appearance representation is that it contains spatial information about the appearance. However, the method gives problems for objects that change poses, since the spatial information then is no longer the same. Instead of templates, histograms of simple shapes might be used.

- **Active appearance models**. These models takes both appearance and shape into account. In the training phase, a model of the shape and the appearance is generated of the target object. When representing an object, the parameters that best make the model match the specific object, are estimated.

- **Multiview appearance model**. These models include different views of an object. One way of representing these views is to generate a subspace from the given views, e.g. using Principal Component Analysis (PCA). The appearance in all views must be available ahead of time.

Feature selection

In this subsection, the different features used to describe objects are investigated. There is usually a close connection between the visual representation of the objects and the features selected for tracking the objects. Four groups of commonly used features are listed and described below.

- **Color.** In image processing, the RGB space is often used. However, the RGB color space has properties that are undesirable in tracking. For example, since the R, G and B components are all correlated with the amount of light hitting the object in sight, they are highly correlated with each other. Other color spaces are therefore also used. This includes the L*u*v, L*a*b, HSV and the RGI color space. These color spaces all have desirable properties. For instance, HSV and RGI separate the color and the light and thus decorrelate the components from the amount of light hitting the object. RGI is a simple and fast color model that also separates the color and light. Conversions from RGB to RGI is faster than from RGB to HSV. [28] However, the L*u*v, L*a*b, the HSV and RGI color spaces are all sensitive to noise. Therefore, it cannot be said, which color space is the optimal to use and thus it depends on the application in question. Color features are all sensitive to changes in illumination.

- **Edges.** Edges are used to track the boundary of an object. A popular approach of detecting the edge of an object is the Canny Edge detector. Edges are less sensitive to changes in illumination than color features.

- **Optical flow.** Optical flow describes the pattern of movement in a scene. More specifically, it is a field of displacement vectors that describes the translation of each pixel in a region. Optical flow features are commonly used in motion-based segmentation and tracking. A popular approach for computing optical flow is the Lucas-Kanade method.

- **Texture.** Texture descriptors express the variation of intensity within a region. There are several texture descriptors. A popular texture descriptor is the Gray-Level Co-occurrence Matrix, which is a 2D histogram that shows co-occurrences of gray-level intensities in a specified direction and distance. Another popular descriptor is the use of Haar-like features that encode spatial intensities using Haar wavelets, as illustrated in Figure 2.6.[29] Another used texture feature is Local Binary Patterns (LBP).[30] Texture descriptors are less sensitive to changes in illumination than color features.

- **Gradients.** Gradients can be used as features to describe texture and edges of an object. The gradients are calculated for each pixel or only for edge pixels in horizontal and vertical directions, which produces a 2D gradient vector. A widely used method for encoding the gradient information is the Histogram of Oriented Gradients (HOG) method, which makes histograms of the gradients in a sliding window in the image. [31] Scale-invariant feature transform (SIFT) is another popular method. It extracts keypoints that are invariant to scale and rotation. These keypoints are, by means of gradients, transformed into a representation that allows for a significant level of local shape distortion and change in illumination. [32] A method similar to SIFT is speeded up robust features (SURF), which is simpler and faster, yet achieving approximately the same performance as SIFT. [33]

Features have usually been chosen manually by the developer, depending on what fits the specific application. A different approach, however is to use automatic feature selection methods. These can be grouped into filter methods and wrapper methods. Filter methods select features using a general criteria (e.g. the features must be uncorrelated), whereas wrapper methods select the features based on how useful they are for a specific problem. An example of filter methods is PCA, which transforms possibly correlated features into a lower dimensional representation, in which the variables (the principal components) are uncorrelated and still express most of the variability in the data. An example of wrapper methods is the AdaBoost algorithm. AdaBoost finds a weighted combination of weak classifiers (each representing a feature), yielding a strong classifier. The classifiers might be trained for one feature each. The n highest weighted features can then be used for tracking. Haar-like features are often used in the Adaboost framework. [29]

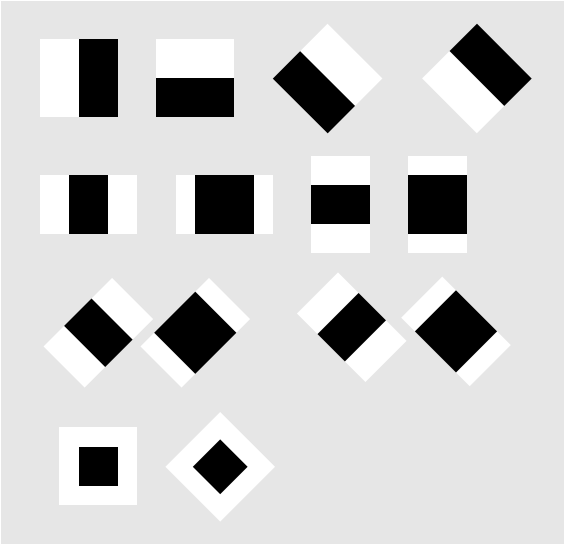


Figure 2.6: *Examples of Haar wavelets that are convoluted with the input image to produce Haar-like features. [34]*

Color is most widely used for tracking, although most color bands are sensitive to changes in illumination. In scenarios, where changes in illumination cannot be avoided, different features are included to represent the object. Furthermore, different features are also combined and used together with the color of the object.

Object Detection

Before the tracking of objects can be done, the objects have to be detected in the image. The detection is either performed based on one frame, or the movement of the object through several frames is used to detect it. When an object has been detected, a tracker must then be able to determine the corresponding positions of the object between frames. This can be done with or without the object detector. Object detection approaches are split into three categories in the following; background subtraction, segmentation and supervised learning.

Background Subtraction A background subtraction method is able to classify each pixel as a foreground or as a background pixel. Since objects that are to be tracked usually are part of the foreground, these background subtraction methods are widely used for object detection.

Before a pixel can be classified, the background must be modeled. This can be done in a number of different ways. One way is to model the noise from the pixel sensors by using a Gaussian distribution to represent each of the pixels. The distribution uses two parameters for each pixel x,y: a mean (μ_{xy}) and a variance (σ^2_{xy}) that are estimated over a sequence of images. For every pixel in an input image, the probability of the pixel coming from the estimated distribution is calculated. If the probability is sufficiently high, the pixel is considered to be a background pixel. Otherwise, it is considered a foreground pixel.

Using one Gaussian distribution is not always sufficient. This can for example be when dealing with outdoor scenes. Here, a pixel can have a multi-modal distribution since the background might be unsettled. In this case, researchers have used mixture of Gaussians, which consist of more than one Gaussian distribution for each pixel in the model. The background might change over time, caused by for example change in lighting. Therefore, the model of the background has to be updated. This has been done by continuously updating the parameters of the Gaussian distributions in the mixture of Gaussian. This is done by evaluating the color value of the pixels compared to the mixture of Gaussian model.

Another background subtraction method, which has been used in research, is background subtraction done in a more spatial way. Each pixel is classified based on the corresponding pixel in the model and a region around it. This makes the subtraction more robust against camera jitter and small movements in the background. Other spatial methods splits the input image into blocks of 5×5 pixels and classifies the image per block.

A problem when dealing with background subtraction is change in illumination, since it can make the background pixels in the input image vary from the background model and therefore be classified as foreground. This has been dealt with by performing a PCA on a number of background images with no objects in it at different light conditions. When a subtraction is done by projecting the image into the eigenvector space produced by a PCA, the result is used as reference image for a pixel-wise subtraction.

In research, background subtraction has been widely used for human tracking. However, the result from a background subtracter is often not perfect, which shows as holes inside the objects and false positive objects. Furthermore, background subtraction methods are not suited for applications, where the camera is moving, since such movement makes the background model invalid.

Segmentation Segmentation methods are able to segment an image into a number of regions, which by some criteria resemble each other. A good segmentation should contain the complete object. Researchers have been using image segmentation to track humans.[35]

One method for segmenting an image is to use mean-shift clustering. This is a clustering method of pixels, which is based on both color and spatial information. Each pixel is given a feature vector, which contains the color in the L*u*v color space and its position: $[l, u, v, x, y]$. Initially, a number of pixels are randomly chosen from the image as cluster means, and every other pixels are clustered together with the closest mean in the feature space. New means are calculated for each of the clusters and again every pixel is clustered to the closest mean. This is repeated until the cluster means no longer change. Each cluster is then considered a segment.

Researchers have also represented an image as a graph and used a graph-cut algorithm to segment the image into subgraphs. The image is converted into a graph so that each pixel is a node and the pixels around it is connected with a vertex. The weight between two nodes is typically calculated based on the color of the pixel, the brightness or the texture. Different types of segmentation such graphs have been used in research. One widely used is the minimum-cut algorithm, which seeks to minimize the cut. This means that the sum of the weights in the edge of the segmentation must be minimized.

Supervised Learning When using supervised learning for object detection, a classifier is trained to classify feature vectors extracted from images as one of two classes: object or non-object. This classification uses the features described earlier. The selected features must describe the target object in a way so that a discrimination between the two classes is possible.

In the training phase of a classifier, a number of samples are first extracted and labeled manually. After this, the parameters of the classifier are determined based on the samples, so that the classifier is able to discriminate samples into their corresponding classes. Different classifiers have been used in research. Simple examples of these are Bayes classifiers and k-Nearest Neighbor (k-NN) classifiers.[36] More advanced classifiers include neural networks, decision trees, adaptive boosting (AdaBoost) and support vector machines (SVM). Because of AdaBoost and SVM both being applicable for human tracking, they are described in the following.

AdaBoost is a meta algorithm that iteratively combines a number of weak classifiers into one strong classifier. When training AdaBoost, each training sample gets a weight, which is the cost of the sample being misclassified. The classification errors for each of the weak classifiers are calculated, where a weight is incorporated. At each step, the weak classifier with the smallest classification error is selected and the weights for all the misclassified samples are increased. The algorithm encourages that in the next iteration, the newly selected classifier performs better on the misclassified samples than the previous classifier. AdaBoost is often used together with weak classifiers that use Haar-like features.

SVM classifiers are trained using the labeled samples, where a hyperplane that separates the two classes with the biggest margin is found in the feature space. The two classes are object and non-object. The training samples that are closest to the hyper plane are called the support vectors and are used to describe the plane. Sometimes, a dataset is not linear separable. Therefore, as a part of the SVM method, the feature space can be transformed into a higher dimension, where a linear separation is possible. The functions used for this are referred to as kernels and can for instance be polynomial or Gaussian kernels. The right choice of kernel is application-specific, and thus requires experimental verification.

Object Tracking

Object tracking is the task of locating the position of objects in every frame of a video sequence. Tracking objects across frames is either done separately or jointly. When tracking objects separately, the possible object regions are first found using an object detection algorithm. The correspondence of objects between frames are then handled by the tracker. When tracking objects is performed jointly, the possible region of the object and the correspondence are estimated by iteratively updating the information obtained from previous frames. Different visual representations are appropriate, depending on whether the objects to be tracked are rigid or non-rigid. For rigid objects, eclipses or boxes are often used to represent the objects. For non-rigid objects, silhouette or contour is usually the most appropriate. In this subsection, a number of common approaches for tracking objects are described separately. The state-of-the-art approaches, as described in the next section (section 2.5.2), generally consist of a combination of approaches described in this section. The approaches are put into the categories shown in Figure 2.7.

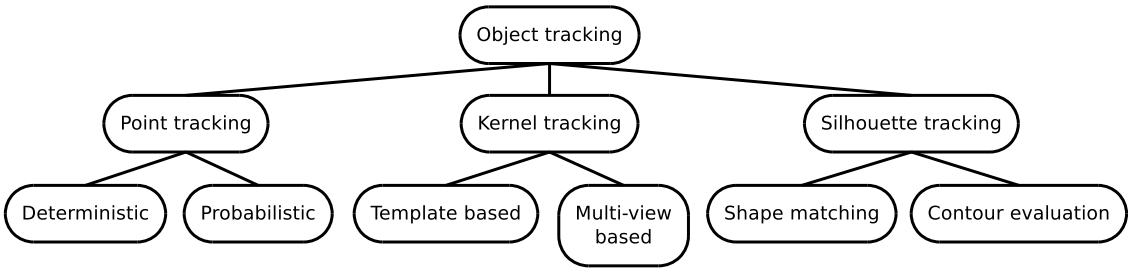


Figure 2.7: *Categorization of tracking approaches.*

The three categories of tracking are now described.

- **Point tracking**. The tracked objects are represented by points. Previous object states that might include position and motion are used for tracking. A method for detecting objects in each frame is required.
- **Kernel tracking**. The objects tracked here are found by using kernels, e.g. templates or elliptical shapes with associated histograms. The motion of kernels across frames is used for tracking the objects. The motion can for example be described by an affine transformation.
- **Silhouette tracking**. For objects with complex shapes, silhouette tracking is often performed. This kind of tracking is done by matching the shapes of the objects, or by evaluating the contours of the objects.

In the following, a number of approaches within the three categories are described.

Point tracking Small objects can be tracked using single points. For larger objects, multiple points are needed. As shown in Figure 2.7, point trackers can be put into two categories; deterministic and probabilistic point trackers. Deterministic point tracking approaches are described first.

Point trackers solve the task of determining one-to-one point correspondence between frames out of all possible associations. Solutions consist of one-to-one correspondences. When using deterministic point tracking, this problem is formulated as an optimization problem. Figure 2.8a shows all possible associations for five points between two frames. Figure 2.8b shows a one-to-one correspondence for the five points.

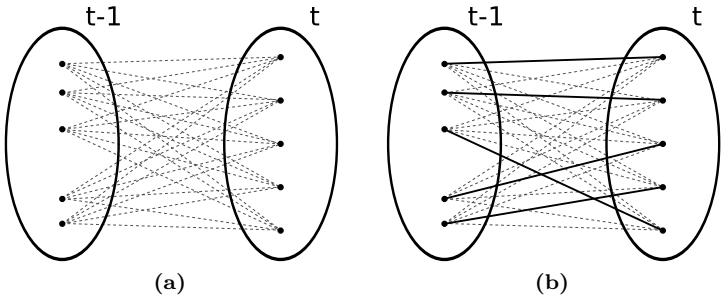


Figure 2.8: *Point correspondence across two consecutive frames. Figure (a) shows the possible associations of five points. Figure (b) shows a solution, i.e. a one-to-one correspondence.*

When finding the optimal assignments, associations between points in t-1 and t are assigned a cost. The problem of minimizing the correspondence cost might be solved using the Hungarian algorithm or by using greedy search algorithms. For a large number of points, investigating all possible associations in order to find the optimal solution will be computationally heavy. Therefore, a number of constraints are usually introduced. These can be assumptions stating that the object does not move significantly between two frames. It should be noted that such constraints are also used in the statistical point tracking methods.

Statistical methods use a measurement and the model of the uncertainties in the measurement to find the optimal assignments. These methods use states of the objects to model their properties. A state might include position, velocity and acceleration of the object. A measurement is usually the position of the object in the frame. For single object state estimation, Kalman filters have been widely used. These assume that the system to be modeled is linear and the state variables of the object are Gaussian distributed. In the case, where these assumptions hold, the Kalman filter provides the optimal solution. To overcome nonlinearity, the extended Kalman filter was proposed, where the system is linearized using Taylor series expansion. The Kalman filter consists of a prediction and a correction step. In the general case, where the state variables are not Gaussian distributed, the Kalman filter will give imprecise estimations. In such cases, a particle filter can be used. Thus, for a particle filter, the system to be modeled does not have to be linear, nor does the state variables have to be Gaussian distributed. Particle filters have steps that are similar to the prediction and correction steps in the Kalman filter. However, as the distributions of the state variables are unknown, they are approximated by samples (particles), where the probability density function (pdf) is modeled and used to estimate corresponding weights. Although particle filters are approximations, they yield a more precise modeling of a non-Gaussian system than Kalman filters.

For multiple object tracking, a joint solution of data association and state estimation is required. The most likely measurement for a specific object must first be associated deterministically to the state of that object. In other words, a one-to-one correspondence must first be found before Kalman or particle filters can be applied. Several methods for this have been used. A simple method is the nearest neighbor method. This method can fail, however, when objects are close to each other. Joint Probability Data Association Filtering (JPDAF) and Multiple Hypothesis Tracking (MHT) are two more widely used methods for data association. A major limitation of JPDAF is that it cannot handle new objects entering the FOV or objects exiting the FOV. Thus, it is only sufficient with a fixed number of objects in the FOV. MHT overcomes this limitation by maintaining several correspondence hypotheses for every object at each frame. Besides its ability to handle objects entering and leaving the FOV, it can also handle occlusions. Since MHT makes

the associations by investigating all possible associations, it can be computationally heavy. To overcome this problem, a probabilistic MHT (PMHT) has been proposed. PMHT reduces the complexity of the MHT by considering the associations to be statistically independent random variables. All in all, for instance when using a particle filter to track multiple objects, measurements might be associated with particular objects through the use of JPDAF, MHT or PMHT. Particle filters that use multiple measurements to track multiple objects have also been proposed.

For point trackers in general, the problem of automatically clustering points that lie on the same object is important. This is because of the need to tell the difference between different objects and between the objects and the background. For this, it is often assumed that the tracked points lie on rigid bodies.

Kernel tracking Kernel tracking approaches fall into one of the two categories: template and density-based appearance models or multiview appearance models. The approaches that use template and density-based models are described first. When using these models for single objects, the most common approach is template matching. The templates are usually generated online, and the position of a template in a frame is computed using a similarity measure like cross correlation. This method is a brute force approach, even though its complexity can be reduced by assuming that the position of a template is in the neighborhood region of its position in the previous frame. Template matching is sensitive to rotation and scale. Another approach is using the KLT tracker, which iteratively computes the translation of a region (a "patch") that is centered on an interest point (a point found to be suitable for tracking). The quality of the tracked patch is evaluated by computing the affine transformation between corresponding patches in two consecutive frames. The tracking continues if the sum of squared difference between the current patch and the patch projected by the affine transformation is small enough.

For tracking multiple objects, the "Layering" approach was proposed. This approach models the entire image as layers. One background layer and then one layer for each object. Each layer consists of shape priors, motion model and an appearance model. Parameters for each object are estimated iteratively using an expectation maximization algorithm. A drawback of this approach is the difficulty of simultaneously estimating the parameters, which means that one set of parameters are estimated at a time, while the others must be fixed. Another approach is "Bramble", which makes a joint modeling of foreground and background regions. A limitation of this approach is that the maximum number of objects in the scene must be predefined.

As the templates used in the aforementioned approaches usually are generated online, the approaches using multiview appearance models use templates that are stored offline. In this way, the tracker can be trained to be able to see the objects from different views. There are techniques that allow for distortions like illumination changes in the templates. These include eigenspace-based similarity computation. Common for these approaches with many templates stored offline is that they are computationally heavy. Another multiview appearance model based approach is the use of SVM, which is trained on positive and negative samples. The positive samples might be images of objects to be tracked seen from different views, whereas negative samples might be images of regions in the background that could be confused with the objects to be tracked. An advantage with SVM is that information about background objects is incorporated into the tracker.

Silhouette tracking Silhouette tracking is used for tracking objects with complex shapes. Two types of silhouette tracking are used; shape matching and contour matching. Shape matching searches for object silhouettes in the current frame, whereas contour tracking uses state space models or direct minimization of an energy function.

Shape matching can be done in a way similar to template matching, where a search is performed by comparing a model of the object silhouette from the previous frame with an object silhouette from the current frame. When using this approach, the motion of the silhouette between two consecutive frames is assumed to be translational. However, the object model, usually an edge map is reinitialized in every frame after the object is located, thus overcoming issues like viewpoint and illumination changes as well as nonrigid object motion. The use of histograms of colors and

edges as object models has been proposed, where the histograms are generated from concentric circles with various radii centered on a set of control points on a reference circle. The resulting color and edge histograms are invariant to rotation, scale and translation.

Contour tracking is either done using state space models or using direct minimization of a contour energy function. The state space models are defined by the shape and the motion parameters of the contour. The state is updated at every frame and this is done so that the a posteriori probability of the contour is maximized. The a posteriori probability is depends on the prior model state and the current likelihood of the model. The likelihood is usually defined in terms of the distance from the contour to observed edges. A contour tracker, where the object state is defined by the dynamics, modeled as a spring of control points has been proposed, where the control points are moved based on the spring stiffness parameters. The new state of the contour is then predicted using a Kalman filter, and the correction step is done based on image observations, i.e. gradients.

When performing contour tracking using direct minimization of an energy function, the minimization is done either by greedy search or by gradient descent. The energy is expressed by temporal information, i.e. either temporal gradient (using an optical flow constraint derived from a brightness constraint) or appearance statistics generated from object and background regions. It has been proposed to compute flow vectors for each pixel inside the object region in a circular neighborhood using a brute force search. After computing these, the contour energy, based on a brightness constraint is evaluated. This process is performed iteratively until the energy function is minimized.

An advantage of silhouette tracking in general is that it can handle many object shapes.

2.5.2 State of the art

In the previous sections an overview of methods for object tracking widely used in research have been described. With this knowledge, three recent approaches that are considered as state of the art are described in this section. These approaches have been chosen with respect to a number of parameters that are relevant for this work. The parameters are listed below, each followed by a description of the parameter.

- Focused on human tracking.
  - Is the approach focused on tracking humans rather than focused on tracking general objects?
- Handling multi target.
  - Is the approach able to handle multiple targets being present in the scene simultaneously?
- Initialization necessary.
  - Does the approach require manual initialization?
- Require scene knowledge.
  - Does the approach require scene-specific knowledge?
- Tracking occluded targets (object occlusion).
  - The ability of the approach to track occluded targets (on a scale from one to three).
- Tracking occluded targets by other targets (inter-object occlusion).
  - The ability of the approach to track targets that are occluded by other targets (on a scale from one to three).
- Thoroughness of test.
  - How thoroughly has the approach been tested (on a scale from one to three)

- Time consumption.
    - Performance of the approach in terms of processing time (on a scale from one to three).

Multi Instance Learning Tracking

The first state of the art approach is a tracking algorithm developed by Babenko et al. called MILtrack.[23] They focus on generic object tracking, where the object to track needs to be initialized by having the user marking it. Their system is only able to follow one target. It requires no scene knowledge.

They have improved the concept of an adaptive appearance model, where the appearance model is normally trained online with positive and negative samples of the object. The positive samples are extracted at the current object tracker position to discriminate the appearance of the object, while negative samples are extracted around the object to discriminate the surroundings. If the tracker position is slightly wrong, it will result in the appearance model drifting away from the appearance of the real object.

Instead, Babenko et al. on use an online Multi Instance Learning (MIL) by sampling a "bag" of positive image patches around and on the tracker location. This ensures that the model is not based on a single image patch, which might be suboptimal. Instead, the algorithm labels a bag of potentially positive image patches and trains the model in a flexible way. Negative samples are also collected to form a negative bag with image patches further away from the tracker location. The appearance features used are Haar-like features consisting of four rectangles and the classification is done using AdaBoost.

Experiments on eight video sequences with the MILTrack method show that it outperforms four tracking methods otherwise considered state of the art in six out of the eight video sequences.[23] In the other two sequences, it performs second best. The experiments show that the approach handles partial occlusions and changes in the appearance and in the pose of the object. In Figure 2.9, six screenshots from one of the video sequences are shown. In the figure, MILTrack and other tracking methods are tested on changes in appearance and partial occlusion.

MILTrack is designed to be a generic object tracking, and it performs well. However, the tracker has not been tested on tracking pedestrians, but the method is able to handle some of the problems in the pedestrian tracking task; occlusion of humans and appearance change due to varying human pose. The tracker has not been proven to be able to handle inter-object occlusion. MILTrack runs at real-time speeds with 25 FPS on a Core 2 Quad desktop machine.

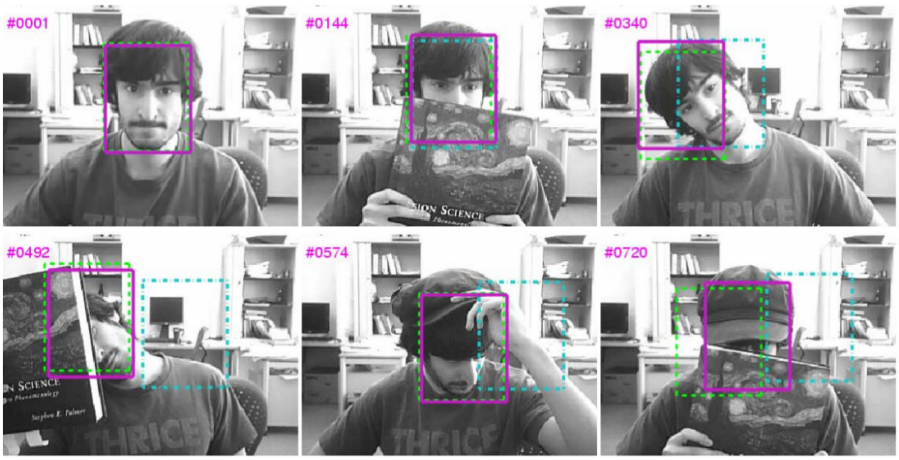


Figure 2.9: *Six frames from one of the test videos for the MILTrack algorithm: Purple solid squares are results of MILTrack, green and cyan dashed squares are the results of the other algorithms (OAB and FragTrack, respectively). Image courtesy of [23].*

Online Learned Discriminative Appearance Model

The second state of the art approach is developed by Kuo et al. [22], where they focus on tracking multiple humans from single cameras and minimizing the number of ID switches when inter-human occlusion occurs. The developed approach does not need any initialization nor any scene knowledge.

The approach is using online learned discriminative appearance model to track the targets. The model is learned based on highly reliable but short tracking segments called tracklets. They are produced using a dual-threshold strategy on association scores between the human detections in two consecutive frames. The association score is based on size, position and color histograms.

The tracklets are used to learn the appearance model by extracting positive and negative samples and are encoded with image features, which are color histograms, covariance matrices and HOGs to form a feature pool. An AdaBoost algorithm is used to weight and thereby vote for the most discriminative features from this pool and to produce a strong classifier. The classifier is then used to determine whether tracklets contain the same target as any other tracklets in the time sliding window.

The approach suggested by Kuo et al. is tested on the CAVIAR and TRECVID08 dataset, which is a dataset with pedestrians in indoor environments, with cameras configured as surveillance cameras. The approach outperforms most of the human tracking approaches otherwise considered state-of-the-art and handles object occlusions. The result of the test showed that the approach is able to generally avoid ID switching when inter-object occlusions occur. The algorithm is implemented in Matlab and runs at 4 fps on a 3.0GHz PC given the output from a human detector. Figure 2.10 shows three output frames from the algorithm tested on the CAVIAR dataset.

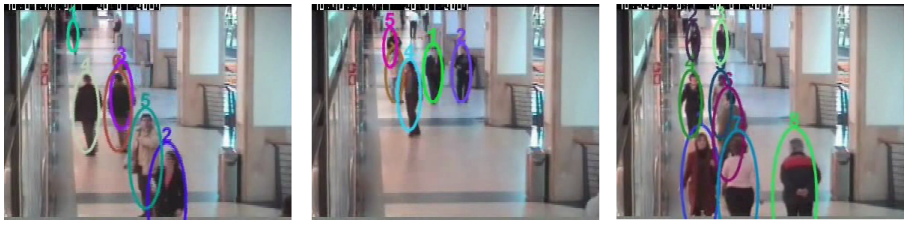


Figure 2.10: *Results of the online learned discriminative appearance model on the CAVIAR dataset. Image courtesy of [22].*

Online Multi-Person Tracking-by-Detection from a Single, Uncalibrated Camera

The third state of the art approach is proposed by Breitenstein et al.[21] They focus on improving the result of available human detectors and tracking individual humans. They do that without any use of specific scene knowledge and with no initialization.

They propose to use a "detector confidence particle filter". The algorithm implements a first order Markov model, using information from the current and the last time step. For each person that is detected with a high confidence, a separate particle filter is initialized. Human detections are used in two ways.

Firstly, high confidence detections are evaluated, and no more than one detection is associated with each target. This association problem is solved by evaluating a sophisticated scoring function that involves run-time trained classifiers and the distance to the tracking target. If this scoring function classifies a detection as being reliable, the detection is used to guide the associated tracker.

Secondly, a continuous "detector confidence" is used together with target-specific classifiers. The detector confidence is used to compensate for erroneous detections and also increase the robustness towards occlusions. It is calculated using the confidence density, which is built up by the human detector. Figure 2.11 shows an output frame from the algorithm.

The algorithm is tested on eight different datasets, and outperforms several other state-of-the art algorithms and handles both object and inter-human occlusions. The algorithm is causal, making it suitable for online applications. It uses a single, uncalibrated camera, and still yields good results when tracking multiple persons in complex scenes with significant occlusions. The

algorithm addresses the problems that are caused by unreliable person detectors and interacting targets. Furthermore, target-specific classifiers are learned at run-time to reduce the number of false-positives. Re-appearing persons are detected, increasing the robustness of the algorithm. The algorithm runs at 0.4-2 FPS on a Intel Core 2 Duo given the output from the human detector.

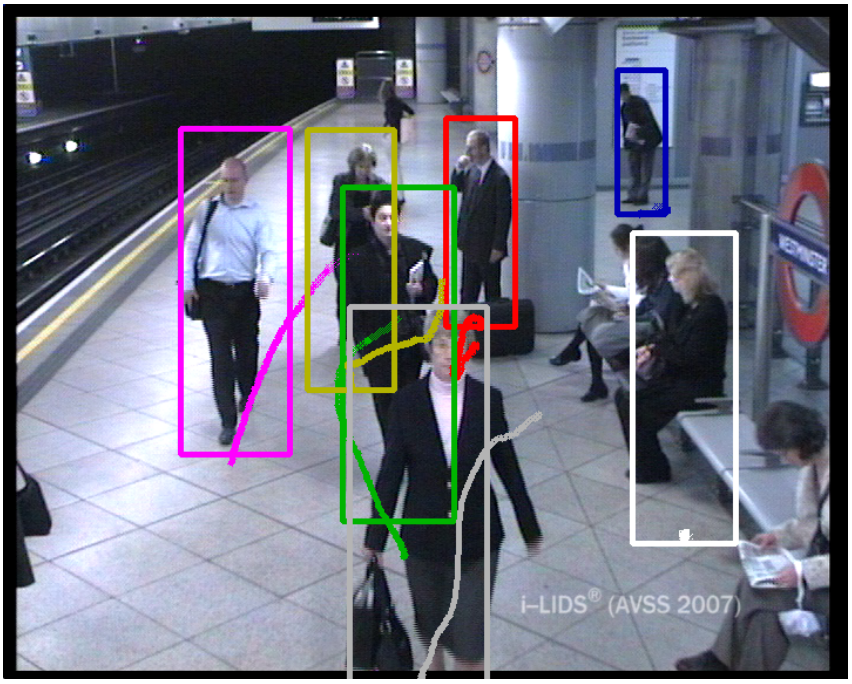


Figure 2.11: *Output frame from the algorithm proposed by Breitenstein et al. Image courtesy of [37].*

2.5.3 Selected Approach

In this section, it is investigated, which of the three state of the art approaces described earlier that best fulfills the need that the system developed in this work would have. I.e. the requirements for a system made for analyzing the movement of people in Friis. For this, they have been evaluated in terms of eight relevant parameters that are defined based on what is necessary of best solving the problem. The evaluation of the approaches in terms of these parameters are summarized in Table 2.2.

All three approaches produce promising results. However, the work by Babenko et al. has not been tested on tracking pedestrians and it has not been proven that this approach is able to handle inter-object occlusion since it only supports single object tracking. The approach did however handle object occlusion well. It needs initialization, but does not use any scene knowledge. It was thoroughly tested on seven different video while tracking different types of objects. The approach gave the best time performance of the three approaches by running at real time speed.

The approach proposed by Kuo et al. is tested on two datasets with pedestrians in indoor environments. However, the experiments showed that the approach is able to track multiple targets and handle inter-object occlusion. Experiments also showed that the approach gave good tracking results without using scene-specific knowledge nor any initialization. It is a vastly time consuming approach and is far from running at real time speed.

The approach by Breitenstein et al. is tested on eight datasets that include datasets with pedestrians and showed that it is able to handle inter-object occlusion, while also producing good tracking results without using scene specific knowledge nor any initialization. This approach is also vastly time consuming and does not run at real time.

Table 2.2 shows a comparison of the three state of the art approaches in terms of the defined parameters.

	Babenko et al.	Kuo et al.	Breitenstein et al.
Focused on human tracking.	No.	Yes.	Yes.
Multi target.	No.	Yes.	Yes.
Initialization necessary.	Yes.	No.	No.
Requires scene knowledge.	No.	No.	No.
Object occlusion.	⋆⋆⋆	⋆⋆	⋆⋆
Inter-object occlusion.	Not tested.	⋆⋆	⋆⋆
Thoroughness of test.	⋆⋆⋆	⋆	⋆⋆
Time consumption.	⋆⋆⋆	⋆	⋆

Table 2.2: *Comparison table for the three state-of-the art approaches for object tracking. The approaches are evaluated on eight different parameters.*

Kuo et al. and Breitenstein et al. have both shown their approaches to be able to track multiple humans which is essential for the task of generating statistics on the movement of people in Friis. The approach by Babenko et al. does not focus on tracking pedestrians and is therefore not appropriate. The approach proposed by Kuo et al. is tested on two datasets showing only pedestrians in indoor environments, while the approach by Breitenstein et al. is tested on eight datasets. These eight datasets include pedestrians in both indoor and outdoor environments and also people playing sports, i.e. hockey and soccer. Thus, the approach proposed by Breitenstein et al. is tested more thoroughly and it performs well on all eight datasets. The approaches by Kuo et al. and by Breitenstein et al. perform similarly in terms of the defined parameters. However, the approach by Breitenstein et al. is tested more extensively. For this reason, the approach proposed by Breitenstein et al. is chosen for this work.

2.5.4 Modifications of Selected Approach

To improve the selected approach, a number of modifications are made. These modifications are made to meet the demand from Friis to generate statistics on the movement of people and also to improve the runtime performance of the system in terms of error rate and time consumption. These modifications are described in the following.

In the previous sections, it was decided to detect and track humans using the method proposed by Breitenstein et al. [21] This method has a number of benefits. These include:

- It can handle complex scenes.
- It is robust against occlusion.
- It needs no initialization.
- It is general, i.e. it requires no scene knowledge or initialization.

The method is shown to yield a low error rate when tested on several widely used datasets. However, when applied for this work, it has a number of areas that need improvement. One problem of the method is its runtime performance. It processes 0.4-2 FPS, given the output from the human detector. For this, the scene information from Friis might be estimated and used to reduce the processing time and reduce the error. Furthermore, the approach is extended to not only estimate trajectories, but to also generate statistics from these.

The modifications of the selected tracking approach are listed below.

- Estimate scene information to improve system performance in terms of:
  - Error rate.

    - Time consumption.

- Generate statistics on the movement patterns of people.
    - Make a visual representation of the movement statistics.

2.6 Problem Description

As described earlier, the managers from Friis are interested in a system that can automatically generate statistics on the movement of their customers using the surveillance cameras. For this, the use of surveillance and the problem of representing the movement of people statistically were analyzed and the related work within the area was thoroughly investigated. The system developed in this work is based on this analysis. In this section, the problem is defined specifically and a number of limitations are set up. Lastly, the analysis is concluded by a problem statement.

In the previous section, "Modifications of Selected Approach" it was stated that the system must generate statistics on the movement of people. The chosen way to solve this is described in the following.

This work focuses on the environment inside Friis. Based on video from the surveillance cameras inside Friis, the system will be able to generate the movement statistics described in Section 2.4, "Movement Statistics". The system must be set to process a video stream, where it stores the trajectories of the persons in the scene. After this, the system must be able to generate movement statistics based on these trajectories, and it must be able to make a visual representation of these statistics.

2.6.1 Limitations

The limitations of this work are listed in the following.

- *No intermediate output:* statistics are generated over the entire time period, i.e. no intermediate output (decribed in Section 2.4) is generated.
- *Single camera:* the system processes video from a single camera at a time.

2.6.2 Data

In this section, the data used in this work is described. This work deals with two scenarios; the Friis scenario and the AAU scenario. Table 2.3 gives an overview of the data.

The Friis sequences are footage from a surveillance camera in Friis. Figure 2.12a shows an image from a Friis sequence. The AAU sequences were recorded specifically for this work at Aalborg University with seven persons participating. Figure 2.12b shows an image from an AAU sequence.

In the AAU4 sequence, the participants were instructed to dwell in front of the glass showcase in the left of the image and at the pole in the top left part of the image. Furthermore, they were instructed to speed up at the top right part of the image. In the AAU5 sequence, the participants were instructed to move in a specific pattern shown in Figure 2.13 at a constant speed, while two participants were instructed to move around arbitrarily.

These instructions that the participants followed are used for verifying the plots generated by the system qualitatively in the Acceptance Test.

Sequence	Frames	GT	Location	FPS	Used for
AAU1	500	Yes.	Aalborg University.	25	Module experiments.
AAU2	2,000	No.	Aalborg University.	25	Subsystem Test; "Preprocessing.
AAU3	127	Yes.	Aalborg University.	25	Subsystem Test; "Processing", "Visualization", Acceptance Test.
AAU4	8,134	No.	Aalborg University.	25	Acceptance Test; qualitative.
AAU5	7,534	No.	Aalborg University.	25	Acceptance Test; qualitative.
Friis1	1,000	Yes.	Friis.	5-8	Module experiments.
Friis2	2,000	No.	Friis.	5-8	Subsystem Test; "Preprocessing.
Friis3	121	Yes.	Friis.	5-8	Subsystem Test; "Processing", "Visualization", Acceptance Test.
Friis4	20,936	No.	Friis.	5-8	Acceptance Test; qualitative.

Table 2.3: *Overview of the data used in this work. The third coloumn states, whether associated ground truth for a sequence has been made.*

(a) **(b)**

Figure 2.12: *Images from the sequences used in this work. Figure (a) shows an image from a Friis sequence. Figure (b) shows an image from an AAU sequence.*

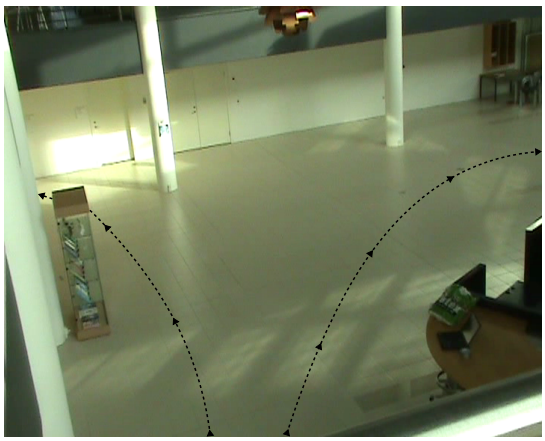


Figure 2.13: *Illustration of the paths, which the participants were instructed to walk along in the AAU5 sequence.*

2.6.3 Problem Statement

The analysis leads to the following goal. The goal for this work is:

To develop a system that can generate statistics on the movement of people inside Friis with a low error rate using a confidence particle filter. The system must estimate scene information to improve the performance of the system in terms of error rate and time consumption. Lastly, the system must make a visual representation of the movement statistics.

Chapter 3

Requirement Specification

In this chapter, the requirements for the system are specified. These requirements are based on the Analysis in Chapter 2. These are the requirements that will be verified in the Acceptance Test in Chapter 12. The statistics that are represented by the system are described in Section 2.4, "Movement Statistics". The requirements for the system are set up in the following. Both a quantitative test and a qualitative test of the system is performed. The qualitative test is performed in order to complement the quantitative test, as the amount of data for the latter is limited due to the time cost of creating GT data.

3.1 Quantitative

For a number of the statistics, their errors are measued as percentage errors. To overcome the problem of zeros being present in the estimated data or the GT data, the Symmetric Mean Absolute Percentage Error (SMAPE) is used for calculating the error.[38] In cases, where zeros occurs in both the estimated and the GT data, the error is defined to be zero percent. The following requirements for the quantitative evaluation of the system are set up:

1. Present heat maps of the following statistics (described in Section 2.4):
    a) Person Position with a mean error less than 0.1.
    b) Speed with a SMAPE less than 15%.
    c) Dwell Analysis with a mean error less than 0.1.
2. Present vector plots of the following two statistics (described in Section 2.4):
    a) Mean Directions;
        1. with a mean angle error less than 10°.
        2. with a magnitude SMAPE less than 15%.
    b) Top-Four Directions with a SMAPE less than 15%.
3. Present a Number of Persons plot (described in Section 2.4) with a mean error less than 1.5.

3.2 Qualitative

The following requirements for the qualitative evaluation of the system are set up:

1. Present reliable heat maps of the following statistics, as described in Section 2.4:
    a) Person Position.
    b) Speed.
    c) Dwell Analysis.
2. Present reliable vector plots of the following two statistics, as described in Section 2.4:
    a) Mean Directions.
    b) Top-Four Directions
3. Present a reliable Number of Persons plot, as described in Section 2.4.

Chapter 4

Acceptance Test Specification

In this chapter, a specification for the Acceptance Test (performed in Chapter 12) is defined. This specification is based on the requirements from the Requirement Specification in Chapter 3. In the following, each requirement is listed along with an instruction written in italic of how the requirement is verified.

Prior to extracting the movement statistics, the scene information in AAU and Friis is estimated:

- For the AAU scenario, this is done by making the "Preprocessing" subsystem process the AAU2 sequence.
- For the Friis scenario, the "Preprocessing" subsystem is set to process the Friis2 sequence.

The requirements are verified both quantitatively (on AAU3 and Friis3) and qualitatively (on AAU4, AAU5 and Friis4). The quantitative acceptance test is specified in the following.

4.1 Quantitative

For visualising the movement statistics, sequences are processed once, and the plots are then evaluated separately. The percentage errors are calculated as SMAPE, expressed in Equation 4.1.[38]

$$SMAPE = \frac{1}{N} \cdot \frac{|\hat{y}_i - y_i|}{\hat{y}_i + y_i} \quad [\%] \quad (4.1)$$

where:

$\hat{y}_i$:	Estimated value of element i.	$[\cdot]$
y_i:	Ground Truth value of element i.	$[\cdot]$
N:	Number of elements.	$[\cdot]$

The following sequences are used for the quantitative evaluation of the system:

- For the AAU scenario: the system is set to process the AAU3 sequence, using the AAU scene information.
- For the Friis scenario: the system is set to process the Friis3 sequence, using the Friis scene information.

To generate ground truth for the movement statistics, the "Visualization" submodule is set to process the ground truth trajectories for the AAU3 and Friis3 sequences. Ground truth for the movement statistics will be referred to as the GT stats from here on. The plots from AAU and Friis are now evaluated separately.

1. Present heat maps of the following statistics, described in Section 2.4:
   a) Person Position with a mean error less than 0.1.
      - *The Sum of Absolute Differences (SAD) between the generated heat map and the corresponding GT heat map is divided by the number of heat map bins, yielding the mean error.*
   b) Speed with a SMAPE less than 15%.
      - *The SMAPE of the generated heat map is calculated as expressed by Equation 4.1.*
   c) Dwell Analysis with a mean error less than 0.1.
      - *The SAD between the generated heat map and the corresponding GT heat map is divided by the number of heat map bins, yielding the mean error.*
2. Present vector plots of the following two statistics, described in Section 2.4:
   a) Mean Directions;
      1. with a mean angle error less than 10°.
         - *The SAD of angles between the generated vector plot and the corresponding GT vector plot is divided by the number of vectors, yielding the mean angle error.*
      2. with a magnitude SMAPE less than 15%.
         - *The SMAPE of the generated heat map is calculated as expressed by Equation 4.1.*
   b) Top-Four Directions with a SMAPE less than 15%.
      - *The SMAPE of the generated heat map is calculated as expressed by Equation 4.1.*
3. Present a Number of Persons plot (described in Section 2.4) with a mean error less than 1.5.
   - *The SAD between the generated plot and the GT plot is divided by the number of frames, yielding the mean error.*

The qualitative acceptance test is specified in the following.

4.2 Qualitative

For visualising the movement statistics, sequences are processed once, and the plots are then evaluated separately.

- For the AAU scenario, the system is set to process the AAU3, AAU4 and AAU5 sequences, using the AAU scene information.
- For the Friis scenario, the system is set to process the Friis3 and Friis4 sequences, using the Friis scene information.

The plots from AAU and Friis are now evaluated separately.

1: Present reliable heat maps of the following statistics, as described in Section 2.4:
   a) Person Position.
      - *The generated AAU3/Friis3 heat map is compared to its corresponding GT heat map.*

- *The generated AAU4/AAU5 heat map is evaluated with respect to the instructions given to the participants in the video, as described in Section 2.6.2.*
- *The generated Friis4 heat map is evaluated with respect to manual inspections of the Friis4 sequence.*

b) Speed.

- *The generated AAU3/Friis3 heat map is compared to its corresponding GT heat map.*
- *The generated AAU4/AAU5 heat map is evaluated with respect to the instructions.*
- *The generated Friis4 heat map is evaluated with respect to manual inspections.*

c) Dwell Analysis.

- *The generated AAU3/Friis3 heat map is compared to its corresponding GT heat map.*
- *The generated AAU4/AAU5 heat map is evaluated with respect to the instructions.*
- *The generated Friis4 heat map is evaluated with respect to manual inspections.*

2: Present reliable vector plots of the following two statistics, as described in Section 2.4:

a) Mean Directions.

- *The generated AAU3/Friis3 vector plot is compared to its corresponding GT plot.*
- *The generated AAU4/AAU5 vector plot is evaluated with respect to the instructions.*
- *The generated Friis4 vector plot is evaluated with respect to manual inspections.*

b) Top-Four Directions.

- *The generated AAU3/Friis3 vector plot is compared to its corresponding GT plot.*
- *The generated AAU4/AAU5 vector plot is evaluated with respect to the instructions.*
- *The generated Friis4 vector plot is evaluated with respect to manual inspections.*

3: Present a reliable Number of Persons plot, as described in Section 2.4.

- *The generated AAU3/Friis3 plot is compared to its corresponding GT plot.*
- *The generated AAU4/AAU5 plot is evaluated with respect to the instructions.*
- *The generated Friis4 plot is evaluated with respect to manual inspections.*

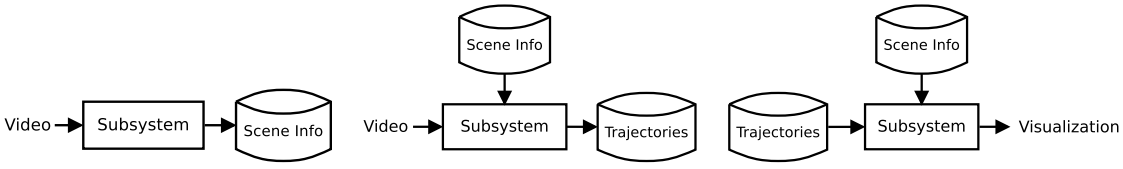
Chapter 5

Program Design

In this chapter, the program design of the system is described. This is based on the Analysis in Chapter 2 and on the requirements that were set up in the Requirement Specification in Chapter 3. The program design consists of a definition of the context, in which the system must work. This is followed by an overview of the proposed method.

5.1 System Context

The system is divided into three subsystems; ”Preprocessing”, ”Processing” and ”Visualization”. These three submodules are run sequentially. The ”Preprocessing” subsystem estimates information about the scene based on a, preferably long video stream. This has to be done once for every camera. The ”Processing” subsystem estimates trajectories of humans in the scene, using the scene information to reduce the complexity of this task. The ”Visualization” subsystem extracts movement statistics from the trajectories and makes a visualization of these. The external interfaces of the subsystems are shown in Figure 5.1.



(a) *”Preprocessing”.* **(b)** *”Processing”.* **(c)** *”Visualization”.*

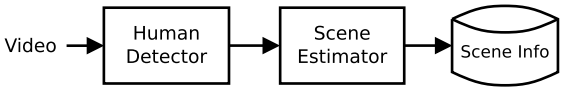
Figure 5.1: *The external interfaces of the subsystems. Figure (a) shows the interfaces of the ”Preprocessing” subsystem, Figure (b) shows the interfaces of the ”Processing” subsystem and Figure (c) shows the interfaces of the ”Visualization” subsystem.*

The system is executed on a PC with an Intel Core2 Duo T9300 2.50GHz CPU and 2GB of RAM. All tests of the system, i.e. the module tests, the subsystem test and the acceptance test are performed on this PC.

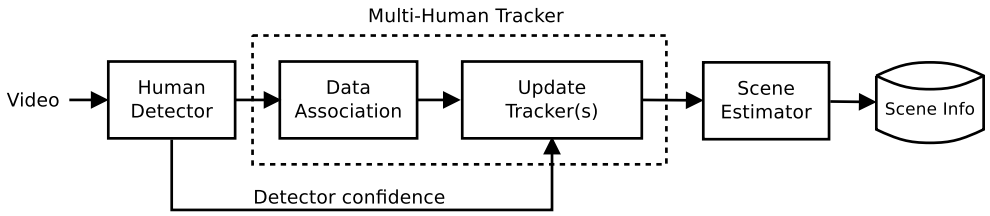
5.2 Proposed Method

Proposed methods for the three subsystems are presented in the following as block diagrams, where each block represents a module. The proposed methods for the ”Preprocessing” (complex solution) and ”Processing” subsystems are derived from the work by Breitenstein et al. and expanded with the ability to estimate and exploit scene information.[21] Figure 5.2 shows an overview of two proposed methods for the ”Preprocessing” subsystem. Figure 5.2a shows a method, where

the output from a "Human Detector" is used to estimate scene information. Figure 5.2b shows a more complex method, where trajectories of humans are used to estimate scene information. Both solutions are developed in order to determine, which estimates the most accurate scene information.



(a) *"Preprocessing"; simple solution.*



(b) *"Preprocessing"; complex solution.*

Figure 5.2: *Overview of two proposed methods for the "Preprocessing" subsystem. Figure (a) shows a proposed solution, where the output from a "Human Detector" is used to estimate scene information. Figure (b) shows a more complex solution, where trajectories of persons are used to estimate scene information.*

Figure 5.3 shows an overview of the "Processing" subsystem, where scene information from the "Preprocessing" subsystem is used to reduce the complexity of the "Human Detector" module.

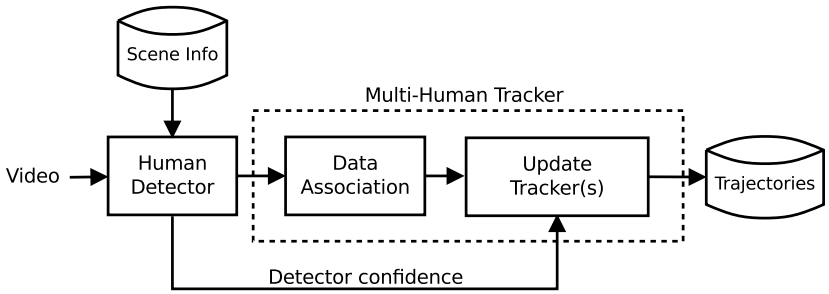


Figure 5.3: *Overview of the "Processing" subsystem. Here, the scene information estimated in the "Preprocessing" subsystem is used to reduce the complexity of the "Human Detector" module. The output is a data structure representing the trajectories of humans in the scene.*

Figure 5.4 shows an overview of the "Visualization" subsystem. Here, the trajectories from the "Processing" subsysten are used together with the scene information from the "Preprocessing" subsystem to make a visualization of the trajectories.

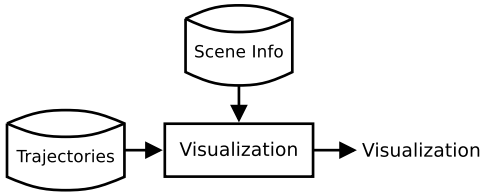


Figure 5.4: *Overview of the "Visualization" subsystem.*

5.2.1 Human Detector

This module detects humans in the scene and outputs the humans positions and size. Intermediate output is sent to the Update Tracker(s) module in the form of a map of raw detector output. This express areas in the frames which looks like humans, but not enough to be an actual detection. The detector does not have to be perfect by detecting all humans in all frames, since the tracker module is able to predict the position of targets in frames with out detections. However, it is more important that this module has a low false positive rate, since false positives can confuse the trackers or cause tackers without targets to be initialized. This module is described in Chapter 6.

5.2.2 Multi-Human Tracker

This module tracks individual humans using particle filters. When tracking using a particle filter, it must first be decided, which detection that should be used for updating which tracker. This is handled by the Data Association module. If the detection can not be associated with any tracker a new target must have entered the scene and the module initialize a new tracker. Updating the trackers with an associated detction is done by the Update Tracker(s) module. The particle filter algorithm is described in Appendix D. Although both being part of the "Multi-Human Tracker" module, the design of the Data Association and Update Tracker(s) parts are described in separate chapters as separate modules. However, the test of the "Multi-Human Tracker" is conducted as one complete test, and is described in the Update Tracker(s) module. Breitenstein et al. evaluate their tracker using the CLEAR MOT metrics, and report Multi-Object Tracking Accuracies (MOTA) varying from 50% to 85.7%.[21][39] The CLEAR MOT metrics are described further in the Experiment section of the "Update Tracker(s)" chapter, where the test of the "Multi-Human Tracker" module is conducted, see Section 8.3.

Data Association

This module part determines, which detection that should guide which tracker by associate a detection to the one tracker which it matches. If a detection has no matching tracker, the detection must represent a new target in the scene and the module must then initialize a new tracker. Furthermore, the module is able to terminate trackers for targets no longer present in the scene. The module is described in Chapter 7.

Update Tracker(s)

Updating the states of the tracked humans is done here. The module is described in Chapter 8.

5.2.3 Scene Estimator

To improve the runtime performance of the system, scene information can be incorporated. The "Scene Estimator" module estimates the scene based on the output from the "Preprocessing" submodule. This module is described in Chapter 9.

5.2.4 Visualization

This module converts a collection of trajectories into a formatted representation from which statistics are generated, and represents the statistics visually. This module is described in Chapter 10.

5.3 Module Specification

In the following, the requirements for the modules are specified. These requirements will be verified in the module tests. The tests of the individual modules are described in their corresponding chapters. All module tests are performed on the AAU1 and Friis1 datasets. The requirements for the modules are as follows.

M.1 **Human Detector**. This module must:

a) Be able to detect humans with a recognition rate of min. 50%.

b) The false positive rate must be less than 5%.

c) Time consumption must be lower than 750ms.

M.2 **Multi-Human Tracker**. This module must:

a) When provided with 50% of all detections in the sequence as ground truth:

1: Track people with an avg. MOTA of min. 60%.

2: Track people with a False Positive (FP) rate less than 20%.

b) When provided with 75% of all detections in the sequence as ground truth:

1: Track people with an avg. MOTA of min. 70%.

2: Track people with an FP rate less than 15%.

c) When provided with 100% of all detections in the sequence as ground truth:

1: Track people with an avg. MOTA of min. 80%.

2: Track people with an FP rate less than 10%.

d) Comply with the following relation between average time consumption, t_c and the number of targets:

$$t_c < 10 \text{ for } n = 0 \quad [\text{ms}]$$
$$t_c < 250 + (n-1) \cdot 150 \text{ for } n \geq 1 \quad [\text{ms}]$$

where:

t_c: Average time consumption of the "Multi-Human Tracker" module. [ms]
n: Number of targets being tracked. [·]

M.3 **Scene Estimator**. This module must:

a) Generate a model that estimates the human scale[2] as a function of image coordinates, with a Root Mean Squared Error (RMSE) lower than 0.15.

M.4 **Visualization**. This module must:

a) Present heat maps of the following statistics, described in Section 2.4:

1. Person Position with a mean error less than 0.1.
2. Speed with a SMAPE less than 15%.
3. Dwell Analysis with a mean error less than 0.1.

b) Present vector plots of the following two statistics, described in Section 2.4:

1. Mean Directions;
   1. with a mean angle error less than 10°.
   2. with a magnitude SMAPE less than 15%.
2. Top-Four Directions with a SMAPE less than 15%.

c) Present a Number of Persons plot (described in Section 2.4) with a mean error less than 1.5.

[2] The "human scale" is defined in the "Human Detector" module, see Chapter 6.

5.4 Subsystem Test Specification

In this section, the specifications for testing the subsystems quantitavely are defined. The test of the subsystems is performed in Chapter 11.The subsystem tests are performed on the AAU2, AAU3, Friis2 and Friis3 datasets.

S.1 **Preprocessing**. This subsystem must:

a) Generate a model that estimates the human scale as a function of image coordinates. The largest distance between the model and the ground truth model must be less than one standard deviation of the data.

S.2 **Processing**. This subsystem must:

a) Track people with an avg. MOTA of min. 50%.

b) Track people with an FP rate less than 15%.

d) Comply with the following relation between average time consumption, t_c and the number of targets being tracked:

$$t_c < 1000 + n \cdot 200 \text{ for } n \geq 0 \quad \text{[ms]}$$

A test of the "Visualization" subsystem corresponds to an acceptance test of the system. Furthermore, the "Visualization" subsystem consists of only one module, for which a module test is performed. Therefore, the test of the "Visualization" subsystem is omitted.

Chapter 6

Human Detector

In this chapter, the "Human Detector" module is described. In Figure 6.1, the placement of this module in the system is illustrated. The task of the module is to detect humans in each frame of the input video sequence, and send it to the "Data Association" module. The module works under two different conditions since the module is used in two different subsystems; the "Preprocessing" and the "Processing" subsystem.

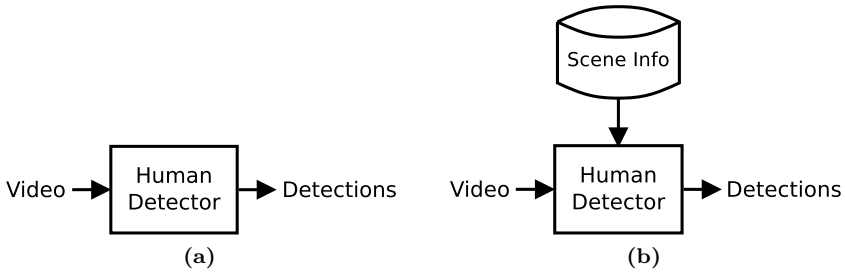


Figure 6.1: *The contexts in which the "Human Detector" module is placed. In Figure (a), module gets a video as input, and outputs human detections. In Figure (b), the module gets a video as input, outputs human detections and uses scene information to reduce the complexity of detecting humans.*

As described in the Module Specification (Section 5.3), the requirements for this module are:

M.1 **Human Detector**. This module must:

a) Be able to detect humans with a recognition rate of min. 50%.
b) The false positive rate must be less than 5%.
c) Time consumption must be lower than 750ms.

In order to detect humans in images, a number of approaches have been used as described in Section 2.5.1. The methods that extract features vectors from image areas and classify them as human or non-human have in recent research been used with success. In [21], the approach used for people detection is HOG features classified using SVM. In this work, the "Human Detector" is based on this approach. Furthermore, the module will exploit the scene information to reduce the search area.

6.1 Module Overview

In Figure 6.2, a diagram shows an overview of the procedure of this module for human detection.

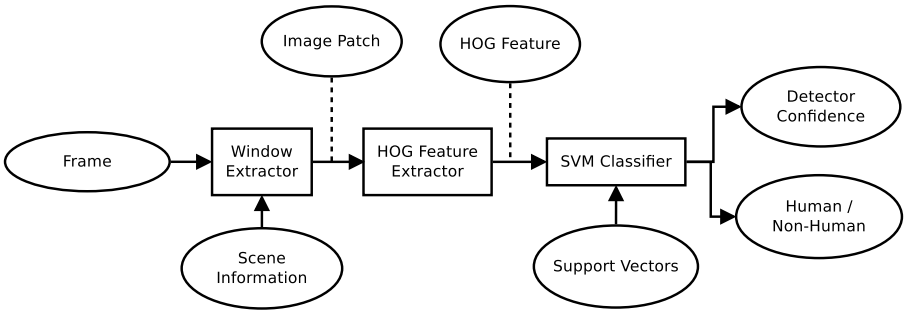


Figure 6.2: *Overview of the "Human Detetion" module. Squares are procedures, while circles are stored information.*

An HOG based detector extracts the HOG features in a sliding detection window across each frame at different scales. This window is extracted from the input frame in the "window extractor". It uses the scene information to reduce the number of windows extracted from the input frame. The scene information is estimated in the "Preprocessing" subsystem. The notion is that the "Human Detector" can save computation time by not attempting to detect people in areas, where the probability of finding people is low. Besides from being used for tracking, the output of this module is also used for estimating the scene information in the "Preprocessing" subsystem, where this module is used unoptimized by searching for humans in the entire frame and at all scales.

The HOG feature extractor calculates a number of normalized histograms of pixel gradient orientations. The feature extraction algorithm is explained in detail in Section 6.3 later in this chapter. The SVM classifier takes this extracted feature and classifies it into two classes: human or non-human. The classification is made based on support vectors that describe the SVM classification and are found in an offline training phase. Furthermore, the SVM classifier outputs the confidence of all extracted windows. This is used to construct a detector confidence map showing, where in the image humans might be but do not yield high enough SVM values to be classified as humans.

6.2 Window Extractor

The "Human Detector" makes use of a sliding window, which extracts image patches systematically from the input frame. The idea is that for every image patch, the HOG feature is extracted and classified as human or non-human. Normally, a sliding window steps forward with one pixel. However, this yields a great amount of image patches that all need to be processed. Dalal and Triggs, the first to use HOG features for people detection propose to make the window step 8 pixel in order to save computation time. This greatly reduces the computation time, while still providing a sufficiently high resolution for the task of human detection. [31]

People have different heights and can stand in different areas of the scene, thus having different sizes in the image. This is illustrated in Figure 6.3. As a consequence of this, the sliding window needs to process the input frames at different window sizes so that people in different heights and in different areas of the image are found. The length of the HOG feature vector extracted from the image patches depends on the size of the patch. A bigger image patch yields a longer feature vector. Since a varying size of the feature vector is not suitable for classification, the vector must be of the same size. This conflicts with a sliding window, which extracts image patches with different sizes. This problem is solved by resizing the input image instead of resizing the window.

The windows size is 64×128 which was found by Dalal and Triggs to perform well. [31] The input image is resized at a number of different scales. The scale of an image, where a human is detected, expresses the size of the human in the unscaled input image. The relationship between a scale and the pixel height of a person in the unscaled image is expressed by the following equation:

$$h_p = \frac{ht_p}{s} \quad [\text{px}]$$

where:

h_p: Pixel height of the person in the unscaled image. [px]
s: Scale of the input image, in which a person is detected. [·]
ht_p: Height of the sliding window (128 pixels). [px]

(a) **(b)**

Figure 6.3: *Illustration of the pixel heights of a person. Figure (a) shows a human far away from the camera. Figure (b) shows the same human close to the camera. Note that a person with the same height has a different pixel height.*

In this work, the "Human Detector" is used for detecting humans, who walk inside a shopping mall. This implies in most cases that the humans are walking on a plane floor, i.e. simple physics can be used to reduce the area for where to search for humans.Thus, humans will have approximately the same size in pixels in a given point of the image. With this intuition a model for the scene can be represented by the size of the people as a function of where in the frame the search is performed. Since all humans do not have the same height, their pixel height at a given position in the frame will also vary. In order to handle this, a margin is introduced in the scene model.

The window extractor uses the scene model to know, when its acceptable to skip an extraction given its position and size. This saves computation time from the feature extraction and the classification. Furthermore, it is able to remove unrealistic big or small detections and thereby reduce the number of false positives.

More details on the scene model and how it is estimated is described in Chapter 9, "Scene Estimator".

6.3 HOG Feature Extractor

In this section, the HOG feature extraction of image patches is described. The HOG feature set was proposed in a paper by Dalal and Triggs, where it outperforms Haar wavelets and SIFT based detectors for human detection. Dalal and Triggs have investigated the performance of HOG features for human detections at numerous internal parameter setups. The parameters used in this work are the ones shown to yield the best result for human detection. [31]

Before the features are extracted from an image, they are gamma normalized. This means taking the square root of each color value for each pixel. The gradients of the image are calculated by convolving a 1D kernel with the rows and columns. The kernel is defined as [1 0 -1] and yields a gradient vector for each color of every pixel in the image.The gradient with the greatest magnitude from the three color channels is selected as the single gradient used for that pixel. Figure 6.4b shows an example of averaged magnitudes of gradients for a set of training samples.

The image patch is divided into a number of rectangular cells, where a histogram of the orientations is calculated for each of these. The histogram consists of nine equally sized bins in the range of 0 to 180 degrees. Each pixel orientation vote is interpolated trilinearly between neighbor bins in order to prevent sudden changes in histograms. This means that the vote is shared between the closest bins weighted by the distance between the bin centers. The size of the each cell is 8×8 pixels. The cells are grouped into blocks of the size 16×16 pixel. Thus, each block contains 4 cells. However, one cell belongs to multiple blocks since the blocks are overlapping each other by the size of one cell. This is illustrated for a small region in Figure 6.4a

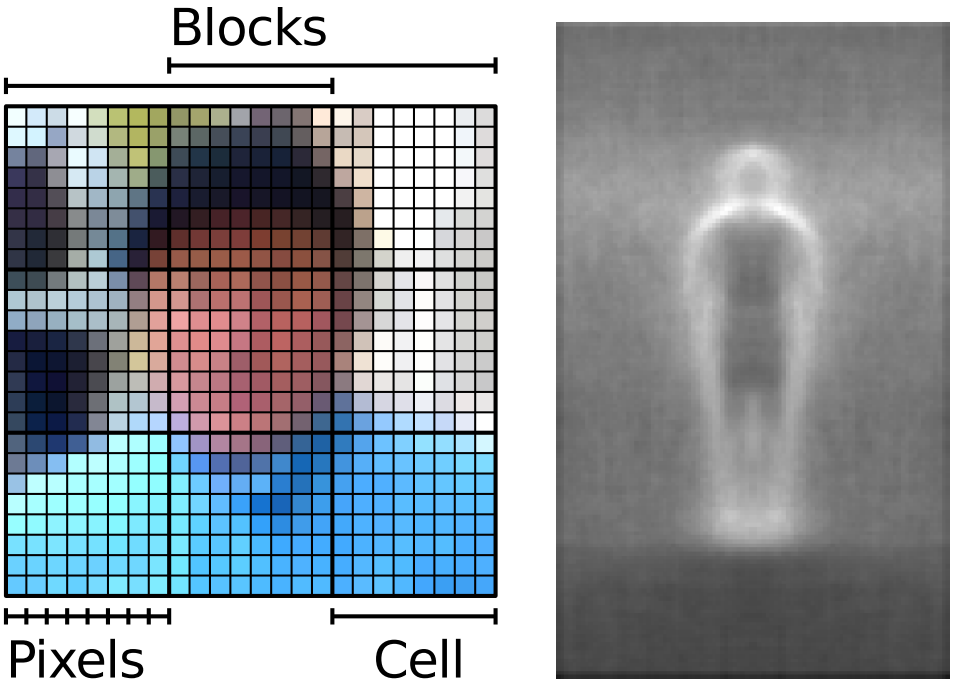


Figure 6.4: *(a): Illustration of the relation between pixels, cells and blocks for a small region in an image. (b): Averaged magnitutes of gradients for all training samples.*

Each contribution of a pixel to the histogram is weighted with a Gaussian kernel, which is centered in the block. This makes pixels far away from the center of the block have smaller impact on the histogram than pixel closer to the center. The four histograms from each block are normalized using an L2-Hys norm. An L2-Hys norm is a normal L2 norm as seen in Equation 6.1 with a clipping limit set to 0.2, i.e. all values above 0.2 in the vector are set to 0.2.

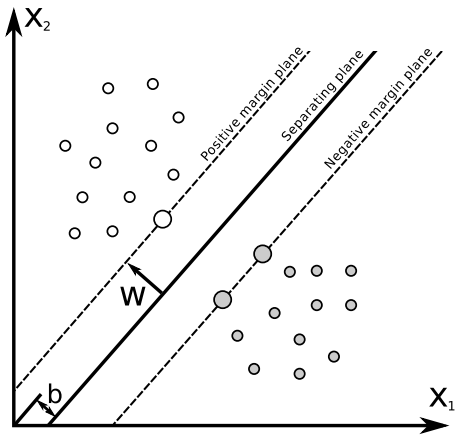


Figure 6.5: *An illustration of the optimal hyperplane (and the margin planes) separating the training data in the feature space with the weight vector* **w** *and the bias b.*

$$\mathbf{x} = \frac{\mathbf{x}}{\sqrt{||\mathbf{x}||^2 + \epsilon^2}} \qquad [\cdot] \quad (6.1)$$

where:

$\mathbf{x}$: A vector containing the four histograms of a block. $[\cdot]$
ϵ: A small constant to avoid zero division. $[\cdot]$

Each image patch contains 7×15 half overlapping blocks, which each consists of 4 histograms each containing 9 bins. This results in a feature vector of an image region that is 3780 elements long.[40]

6.4 SVM Classifier

In this section, the training and classification with the SVM classifier is described. SVM is used to binary classify the HOG feature vector extracted from each image patch. An image patch can be classified into two classes; human or non-human.

An SVM is a supervised binary classifier widely used for object detection. It is trained with a set of samples that have been labeled manually into the two classes mentioned before. A training dataset is given by:

$$S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)) \qquad [\cdot]$$

where:

$\mathbf{x}_i$: The feature vectors of the training data. $[\cdot]$
y_i: The corresponding class of the feature vector. $\in \{-1, 1\}$ $[\cdot]$
n: The number of training samples. $[\cdot]$

The version of SVM called Maximum margin can only be used on data that is linear separable. This means the samples of the two classes can be separated with a linear hyperplane in the feature space. SVM is for such data able to find the best separating hyperplane. That is the plane with the maximum margin to the nearest training samples. These nearest samples are called the support vectors. The best separating plane from a fictional dataset is illustrated in Figure 6.5. A

hyperplane is represented by a weight vector, which is perpendicular to the plane and a bias. The hyperplane is expressed by the following equation:

$$0 = \mathbf{w} \cdot \mathbf{x} - b \quad [\cdot]$$

where:

$\mathbf{x}$: A point on the hyperplane. $[\cdot]$
$\mathbf{w}$: The weight vector of hyperplane. $[\cdot]$
b: The hyperplane bias. $[\cdot]$

To obtain the hyperplane that yields a maximum margin, it must intuitively be placed in the middle of the nearest samples from the two classes. SVM introduces a functional margin γ_i for a given feature point $\mathbf{x}_i$ and its corresponding class y_i as:

$$\gamma_i = y_i(\mathbf{w} \cdot \mathbf{x_i}) - b \quad [\cdot]$$

SVM defines the marginal function of the closest points from both classes in the training set (the support vectors) to be 1. This implies that the hyperplanes of the margins (the dashed lines in Figure 6.5) are defined as:

$$+1 = \mathbf{w} \cdot \mathbf{x}^+ - b \quad [\cdot]$$
$$-1 = \mathbf{w} \cdot \mathbf{x}^- - b \quad [\cdot]$$

where:

$\mathbf{x}^+$: Support vector from the positive sample class. $[\cdot]$
$\mathbf{x}^-$: Support vector from the negative sample class. $[\cdot]$

This results in the geometric distance between the separating plane and the margin plane to be $1/||\mathbf{w}||$ [41, p. 95]. This means that the smaller $\mathbf{w}$ is, the greater the margin of the classifier is. Since the support vectors are the points closest to the separating plane, all other point must have a functional margin greater than 1. This can be written as:

$$y_i(\mathbf{w} \cdot \mathbf{x} + b) \geq 1 \quad [\cdot]$$

Now it is possible to formulate an optimization problem to find the best separating hyperplane of the training data:

$$\begin{aligned} &\textit{minimize}_{\mathbf{w},b} && \mathbf{w} \cdot \mathbf{w}, && [\cdot] \\ &\textit{subject to} && y_i(\mathbf{a} \cdot \mathbf{x}_i + b) \geq 1, && [\cdot] \\ & && i = 1, \ldots, n && [\cdot] \end{aligned}$$

Inequality constrained optimization problems can be hard to solve, and therefore the problem is transformed into a dual form using the Lagrange multipliers. [41, p. 95] The resulting optimization problem is formulated as:

$$\textit{maximize} \quad W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \mathbf{x}_i \cdot \mathbf{x}_j, \quad [\cdot] \quad (6.2)$$

$$\textit{subject to} \quad \sum_{i=1}^{n} y_i \alpha_i = 0, \quad [\cdot]$$
$$\alpha_i \geq 0, i = 1, \ldots, n \quad [\cdot]$$

where:

α: The Lagrange multipliers. [·]

When the Lagrange multipliers are calculated, the weight vector is obtained by:

$$\mathbf{w} = \sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i \qquad [\cdot]$$

The weight vector of the seprating plane can now be estimated from training samples and thus, classification is possible. The feature vector is evaluated in the estimated SVM by calculating the dot product between the weight and the feature vector. The dot product is the confidence of the detector and is used for constructing a confidence map from all the windows extracted. If the confidence is above 0, the windows for which the feature vector is extracted from is considered a human detection.

6.5 Detector Training

The training of the human detector is done on the INRIA dataset, which was produced by Dalal and Triggs. [31] It contains 1805 cropped image of the size 64×128 with humans standing up in different poses and levels of occlusion. The cropped images have 16 pixel of margin in top and bottom with background. Dalal and Triggs discovered that if a training image also contained context, i.e. the background of the humans, the performance increased. [31] As a consequence of this, the detections that the human detector module produces also include a margin of 16 pixels in the top and the bottom of the bounding boxes.

From the 1,805 images in the INRIA dataset, 1,239 images are selected as positive samples for the training phase. Furthermore, a left-right flipped copy of each positive training sample are added to the positive sample pool. 1,2180 negative training samples are extracted from 1,218 photos without humans. The HOG feature vectors are extracted from positive and negative training images and is used to train the SVM classifier.

6.6 Experiments

In this section, the experiment that is performed to evaluate the "Human Detector" module is described. The experiment is performed in accordance with the Module Specification in the Program Design. The experiment is run on the AAU1 and the Friis1 sequence. For both AAU and Friis, the experiment is performed both using and without using scene information. For a detailed description on how the output of the module is compared to the corresponding GT, see Appendix A, "Test and Evaluation Procedure".

M.1 **Human Detector**. This module must:

a) Be able to detect humans with a recognition rate of min. 50%.
Friis: 51.02%.
Friis /w scene info.: 53.98%.
AAU: 57.08%.
AAU /w scene info.: 60.17%.

b) The false positive rate must be less than 5%.
Friis: 31.91%.
Friis /w scene info.: 4.98%.
AAU: 40.79%.
AAU /w scene info.: 1.34%.

c) Time consumption must be lower than 750ms.
Friis: 913.60ms.
Friis /w scene info.: 554ms.

AAU: 1,141.00ms.
AAU /w scene info.: 751.18ms.

6.7 Discussion

The results of the experiment show that the scene information decreases the number of false positives significantly without increasing the number of false positives. Inspections have shown that the false positives often are unrealistic large or small, thus being filtered out using the scene information. Furthermore, the use of scene information decreases the time consumption with approximately 36%.

Thus, the module fulfills the requirements when using scene information except for AAU, where the time consumption exceeds the requirement marginally, i.e. 1.18ms.

Chapter 7

Data Association

In this chapter, the "Data Association" module is described. This module associates the detections from the "Human Detector" module (described in Chapter 6) with the corresponding tracker. If a new person enters the scene, this module is able to detect that and initialize a new tracker for that target. Furthermore, the module is able to terminate trackers for persons leaving the scene. The placement of the "Data Association" module in the system is shown in Figure 7.1.

Figure 7.1: *The context in which the "Data Association" module is placed.*

7.1 Module Overview

In Figure 7.2, an overview of this module is illustrated. First, the module calculates a matching score between each human detection and all the trackers. This score is an expression for the probability of one detection being the target followed by a tracker. The score is not only based on spatial distance between the tracker position and the detection position, but also on the appearance of the target. The appearance is modeled with an online learned AdaBoost classifier, which is individual for each tracker. The scores between all detections and trackers are saved in a scoring matrix. The actual tracker/detection associations are found using greedy search based on the scoring matrix and the trackers are updated with the associated detection.

Detections not yielding high enough score to be associated with any trackers are sent to the initialization/termination part. Here, it is investigated if the detection is a new person who has entered the scene or if it is a false detection. Furthermore, it is investigated if trackers that have no associated detection is caused by the target it tracks have left the scene. If that is the case the tracker should be terminated.

7.2 Scoring Function

The scoring function is a function, which is able to estimate the similarity between a tracker tr and a detection d. Breitenstein et al. propose to base the similarity measure on four parameters: the spatial distance, the difference in size, the appearance, and the motion. The scoring function is given by:

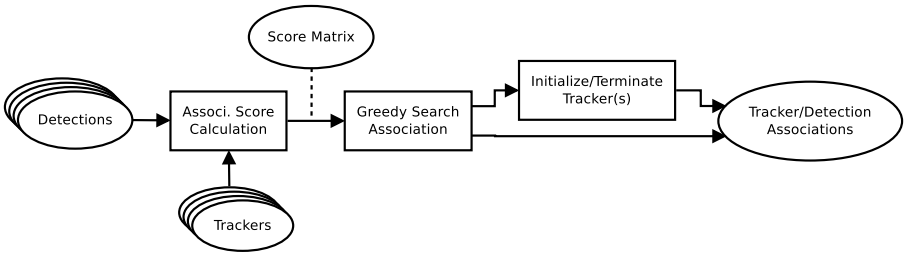


Figure 7.2: *Overview of the "Data Association" module. Squares are procedures while circles are stored information.*

$$S(tr, d) = g(tr, d) \cdot (c_{tr}(d) + \alpha \cdot D(tr, d)) \quad [\cdot]$$

where:

$g(tr, d)$: Gating function. $[\cdot]$
$c_{tr}(d)$: Appearance parameter. $[\cdot]$
$D(tr, d)$: Distance parameter between tracker and detection. $[\cdot]$
α: Weight of the distance parameter. $[\cdot]$

The gating function $g(tr, d)$ expresses the motion and size similarity of the tracker with respect to a detection. The appearance parameter, $c_{tr}(d)$ is based on an online learned classifier using appearance specific features. The distance parameter, $D(tr, d)$ is not only calculated from the center of the tracker to the center of the detection. Since the tracker makes use of a particle filter for estimating the target position, all the particles can be seen as hypotheses for the position of a target at a given time step. All these hypotheses are included in the distance parameter by defining the distance to be the sum of distances between all particles and the detection. All distances are evaluated on a Gaussian distribution. The distance parameter in the scoring function is then defined by Breitenstein et al. as:

$$D(tr, d) = \sum_{p \in tr}^{N} p_{\mathcal{N}}(d - p) \quad [\cdot]$$

where:

$D(\cdot, \cdot)$: The distance parameter between tracker tr and detection d. $[\cdot]$
$p_{\mathcal{N}}(\cdot)$: Zero mean normal distribution. $[\cdot]$
N: Number of particles. $[\cdot]$

For calculating the appearance parameter, $c_{tr}(d)$ the detection is classified by a classifier that has been learned online for each tracker. The classifier is a strong classifier combined by a number of weak classifiers in an AdaBoost framework. The AdaBoost algorithm used by Breitenstein et al. is an online boosting based feature selection framework which was proposed by Grabner et al.[42] The algorithm and the features used are described in Appendix C. The raw output from the classification of a detection is an expression for the similarity in appearance between the detection and the tracker, which the classifier belongs to. This raw output is used in the scoring function.

The classifier is trained with positive and negative samples of a target, but only when a detection has been associated with a tracker. The positive training samples are extracted near the detection on a horizontal line going through the detection center. Negative samples are extracted further away from the detection center and with any other detections there may be. The reason

why the classifier is not updated at every frame with samples extracted from the tracker position is that the position of the tracker may be slightly wrong. This may result in the classifiers being trained on samples that contain more background than the target and decrease the ability for the classifier to classify the target correctly, and thereby making the classifier drift.

To incorporate the motion direction and size of a tracker into the scoring function, a so called gating function $g(tr, d)$ was proposed by Breitenstein et al. The gating function is a product of two parameters:

$$g(tr, d) = p(size_d|tr)p(pos_d|tr) \quad [\cdot]$$

where:

tr, d: A tracker and a detection, respectively. $[\cdot]$
$p(size_d|tr)$: Expression of the difference in size between tr and d. $[\cdot]$
$p(pos_d|tr)$: Expression of the position and motion of tr with respect to d. $[\cdot]$

The first parameter takes the difference in size between the tracker and detection into account. This is done by evaluating a Gaussian distribution on the difference between the tracker and detection size scaled with the size of the tracker.

$$p(size_d|tr) = p_{\mathcal{N}}\left(\frac{size_{tr} - size_d}{size_{tr}}\right) \quad [\cdot]$$

The second term in the gating function, $p(pos_d|tr)$ depends on the speed and direction of a tracker with respect to the detection. It follows the intuition that a fast moving object does not suddenly change direction. This means that detections in front of the direction of the trackers get higher scores than trackers behind them. For calculating the parameter, the orthogonal distance between the detection center and velocity line is found and evaluated on a Gaussian distribution. The velocity line is going through the center of the tracker and has the direction given by the velocity vector of the tracker, $\mathbf{v}_{tr}$ i.e. the direction of the trackers. Furthermore, the variance of the Gaussian distribution is increased as the detection gets further away from the tracker center. This produces a 2D cone in front of the tracker, which is illustrated in Figure 7.3b. If the speed of the tracker is lower than a threshold, the expression only depends on the distance between the tracker and detection evaluated on a Gaussian distribution. The tracker motion/position part $p(pos_d|tr)$ of the gating function is given by:

$$p(pos_d|tr) = \begin{cases} p_{\mathcal{N}}\left(|d - tr|\right) & \text{if } |\mathbf{v}_{tr}| < \tau_v \\ p_{\mathcal{N}}\left(|\text{proj}_{\mathbf{v}_{tr}}(d - tr)|\right) & \text{otherwise.} \end{cases} \quad [\cdot]$$

where:

$\mathbf{v}_{tr}$: velocity vector of the tracker tr. $[\cdot]$
$\text{proj}_{\mathbf{v}_{tr}}(\mathbf{v})$: Is the projection of the vector $\mathbf{v}$ onto the velocity vector. $[\cdot]$

The classifier, gating function and the distance terms all make up the scoring function used together with a greedy search algorithm to associate a detection to one active tracker.

7.3 Greedy Search Algorithm

Breitenstein et al. propose to use a greedy search for data association. As mentioned in the Previous Work section (Section 2.5.1), often used methods for data association include MHT, JPDAF, the Hungarian algorithm and greedy search. The complexity of MHT limits it to only consider a few time steps. For JPDAF, an increasing number of detections results in an exponentially growing complexity. For single frames, the Hungarian algorithm provides the best assignment. However Breitenstein et al. have found through experiments that the greedy search achieves similar results, although at a lower computational cost.

(a) **(b)**

Figure 7.3: *In Figure (a) and (b), the result of the motion/distance part of gating function for a tracker moving at low and high speeds is visualized. Notice how a 2D cone is created in the direction of the trackers movement when it is moving fast.*

A score is calculated using the scoring function described in the previous section between all active trackers and the detections. The greedy search then associates the detection/tracker pair with the highest score, but only if it exceeds a threshold. The associated detection and tracker is removed from the search, and the search for the highest sores is performed again. This is repeated until all tracker/detections pairs are associated or until no one has a score higher than the threshold. The algorithm is described in Algorithm 1.

Algorithm 1 Greedy Search Algorithm

T : set of trackers
D : set of detections
$S(tr, d)$: scoring matrix
$A(tr, d) = 0$: association matrix
while $T \neq \emptyset \wedge D \neq \emptyset$ **do**
  $(tr^*, d^*) = \arg\max_{tr \in T, d \in D} S(tr, d)$
  if $S(tr^*, d^*) > \tau$ **then**
    $A(tr^*, d^*) = 1$
  end if
  $T = T \backslash tr*$
  $D = D \backslash d*$
end while

7.4 Initialization and Termination of Trackers

In the previous sections, it was explained how each detection is associated with only one tracker, in order to update the tracker using the detection. However, the case where a detection is not associated with any trackers or trackers do not have any detections, has not been investigated. Those cases can be caused by a person having entered or exited the scene or caused by false positives or negatives. Therefore, this part of the ”Data Association” module is introduced to determine if a new tracker should be initialized or a tracker should be terminated. The methods used in this work is based on the methods proposed by Breitenstein et al.

7.4.1 Initialization

The simple procedure for initialization of trackers is to initialize every detection, which has no associated tracker. However, the detections can be false positives or the data association can have failed to associate the detection of a target, which already has a tracker. In both cases, a tracker will be initialized with the simple procedure, resulting in multiple trackers following the same target.

Breitenstein et al. propose to increase the confidence of an initialization by requiring that a detection with no association must be confirmed by multiple overlapping detections without associations in previous frames. As a result, the initialization of a tracker is delayed for a couple of frames, but in return it will be more confident. In order to make initializations even more confident, Breitenstein et al. introduce initialization zones in the edges of the frame. These zones define the only areas, where trackers can be initialized. This follows the intuition that people will enter the scene in the edge of the frame. However, this intuition will not always be valid. If the scene has a wall that ends in the middle of the frames, people are able to walk behind it and reappear at the end of the wall, which then might be in the middle of the frame. The people doing this, will not have a tracker initialized.

A tracker in this work is initialized, when a non-associated detection overlaps another non-associated detection in the five previous frames. This increases the confidence for an initialization slightly, while also causing a short delay in the initialization. Furthermore, it makes humans passing through the scene untracked more unlikely.

7.4.2 Termination

For termination of trackers, Breitenstein et al. propose to count the number of frames, in which a tracker has not been associated with a detection. If this count exceeds a threshold, the tracker is terminated. This is a simple method for removing a tracker, whose target has left the scene. Problems with this method might occur if a person enters the scene the in same region, where another recently has left the scene. Then, the tracker might get associated to a detection of the new person and starts to track the new person, resulting in an ID switch. This problem can be minimized by selecting an low threshold for number of frames an tracker can survive without associated detections.

In this work, the threshold for the number of frames in which a tracker can remain active without associated detections is set to 10. This is a relatively high threshold when recalling that the ”Human Detector” module gets more than 50 % of the detections correct according to the experiments of the module (described in Chapter 6). Although the detection rate is 50%, it has been observed that gaps between detection of a target can be greater than the expected gap of one frame. This is normally caused by occlusion by the scene or other targets.

Chapter 8

Update Tracker(s)

This chapter describes the "Update Tracker(s)" module which is part of the "Multi-Human Tracker" module. The purpose of this module is to update the position and size of a tracker. This is done using a particle filtering framework described in Appendix D. This module is based on [21]. Figure 8.1 shows the context in which the "Update Tracker(s)" module is placed.

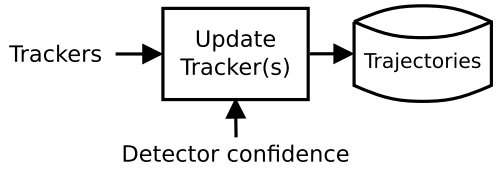


Figure 8.1: *The context in which the "Update Tracker(s)" module is placed.*

This module uses the detections from the "Human Detector" module (see Chapter 6). As multiple trackers and multiple detections might be present at a given time, the detections have been associated with different trackers, i.e., at most one detection has been associated with at most one tracker, as described in Chapter 7. Thus, for the "Update Trakcer(s)" module, a number of the trackers have an associated detection each.

8.1 Design

The chosen technique for updating the position and size of a tracker is described in the following. This module follows a particle filtering framework for updating the position and the size, where the state contains the image coordinates, (x, y) and the velocity components, (u, v). I.e., $\mathbf{x} = \{x, y, u, v\}$. The size of the target is not included in the state. Instead, the size of the target is set to be the average of the last four associated detection. A fixed number of particles is used; $N = 100$.

8.1.1 Iterative Sampling

This module follows the sampling scheme described in Appendix D. For predicting the particles, the following motion model is used.

$$(x, y)_t = (x, y)_{t-1} + \Delta t \cdot (u, v)_{t-1} + \varepsilon_{(x,y)} \qquad [(\text{px}, \text{px})] \quad (8.1)$$
$$(u, v)_t = (u, v)_{t-1} + \varepsilon_{(u,v)} \qquad [(\text{px/s}, \text{px/s})]$$

where:

Δt: Time difference between the current and the previous frame. [s]
$\varepsilon_{(x,y)}$: Process noise for the position. [px]
$\varepsilon_{(u,v)}$: Process noise for the velocity. [(px/s, px/s)]

The process noise for the position and for the velocity are drawn independently from zero-mean normal distributions. The initial values of the variances, $\sigma^2_{(x,y)}$ and $\sigma^2_{(u,v)}$ are set to be proportional to the size of the tracker. Breitenstein et al. propose to use initial variances

$$\sigma_{(x,y)} = 4 \cdot s_{det} \quad [\cdot] \quad (8.2)$$
$$\sigma_{(u,v)} = 12 \cdot s_{det} \quad [\cdot] \quad (8.3)$$

Where:

s_{det}: The scale of the detection. [$\cdot$]

The scale, s_{det}, of the detection is the detection size compared to the size of the training samples. The more frames that the target has been tracked successfully, the more the variances are decreased down to a lower limit. Thus, the longer a target has been tracked successfully, the less the particles are spread. Additionally, [21] use techniques to increase the robustness towards fast camera motion. However, as this work is aimed for surveillance systems, the camera is assumed to be steady.

Initial Conditions

To start the sampling scheme, initial conditions must be set. When a new tracker has been initialized, it only has a position and a size. Thus, its velocity components are zero. Furthermore, the initial variances for the position and velocity noise, $\sigma^2_{(x,y)}$ and $\sigma^2_{(u,v)}$ is defined in Equation 8.2 and 8.3.

Iterating

As the sample set at time $t-1$ has been drifted and diffused, the weights for the samples at time t must be estimated. This is done using the observation model expressed in Equation 8.4.

$$w_{tr,p} = \underbrace{\beta \cdot \mathcal{I}(tr) \cdot p_{\mathcal{N}}(p - d^*)}_{\text{detection}} + \underbrace{\gamma \cdot d_c(p) \cdot p_o(tr)}_{\text{detector confidence}} + \underbrace{\eta \cdot c_{tr}(p)}_{\text{classifier}} \quad [\cdot] \quad (8.4)$$

where:

$w_{tr,p}$: Weight for particle p from tracker tr. [$\cdot$]
β: Weight of the detection term. [$\cdot$]
$\mathcal{I}(tr)$: Function that indicates, whether tr has an associated detection. [$\cdot$]
d^*: Detection associated with tr. [$\cdot$]
γ: Weight of the detector confidence term. [$\cdot$]
$d_c(p)$: Detector confidence density at p. [$\cdot$]
$p_o(tr)$: Inter-object occlusion reasoning for tr. [$\cdot$]
η: Weight of the classifier term. [$\cdot$]
$c_{tr}(p)$: Tracker-specific classifier evaluated for p. [$\cdot$]

The $\mathcal{I}(tr)$ function in Equation 8.4 is an indicator function. It returns 1 if the tracker tr has an associated detection, and 0 otherwise. Thus, the detection term is only included in the calculation of the particle weight if the tracker has a detection. In that case, the detection term evaluates the distance between a particle and the associated detection on a normal distribution. The next

term is the detector confidence term. It evaluates the detector confidence density, $d_c(p)$ at the position of the particle. The detector confidence density corresponds to the raw SVM output from the HOG detector, as described in Chapter 6, "Human Detector".

In certain cases, background structures can cause the detector confidence density to produce values that are too high, making it unreliable in such cases. To increase the reliability of the detector confidence density, inter-object occlusion reasoning is performed by the $p_o(tr)$ function. The rationale behind the inter-object occlusion is that if a tracker has no detection, and no other trackers are close, the detector confidence density should be 0, as it is influenced by background structures otherwise. If a tracker has an associated detection, the detector confidence density is caused by foreground structure. In the case, where a tracker has no associated detection, but is close to another tracker that has an association, the detector confidence density is then most likely caused by the foreground and not the background. This could happen in a situation, where only one of the trackers was detected due to occlusion. Therefore, $p_o(tr)$ increases the detector confidence density as a tracker gets closer to another tracker. The inter-object reasoning function, $p_o(tr)$ is expressed in Equation 8.5.

$$p_o(tr) = \begin{cases} 1 & \text{if } \mathcal{I}(tr) = 1 \\ \max\limits_{tr':\mathcal{I}(tr')=1} p_{\mathcal{N}}(tr - tr') & \text{else if } \exists \mathcal{I}(tr') = 1 \\ 0 & \text{otherwise.} \end{cases} \quad [\cdot] \quad (8.5)$$

where:

tr: Tracker, for which the function is evaluated. $[\cdot]$

tr': The closest tracker close to tr. $[\cdot]$

$\mathcal{I}(tr)$: Function that indicates, whether tr has an associated detection. $[\cdot]$

Figure 8.2 shows a visualization of the inter-object reasoning function.

Figure 8.2: *Visualization of the inter-object reasoning function.*

8.1.2 Output

The output of this module is the position and the size of a target along with a timestamp and an ID of the target. A weighted mean of the particle positions is used to estimate the position of the target. The size of the target is estimated as the average of the last four associated detections.

8.2 Tracker Parameters

In this section, the parameters of the trackers are set experimentally. The parameters are the weights of the three terms in the observation model (β, γ, η).

8.2.1 Weights

For setting the values of the weights in the obervation model, [21] set β, γ and η so that the ratio between the three terms of the observation model are approximately $20:2:1$ for a tracker with an associated detection. In correspondence with [21], the values of β, γ and η will first be estimated by evaluating the three terms in the observation model at a number of frames for trackers with an associated detection. The weights will then be set so that the ratio between the terms are approximately $20:2:1$.

Table 8.1 lists the results from the experiment, where weights in the observation model are set to 1, i.e. $\beta = 1$, $\gamma = 1$ and $\eta = 1$.

Term	Mean value (AAU)	Mean value (Friis)	Mean Ratio
Detection.	0.2215	0.1256	20.00
Det. conf.	0.1421	0.1444	16.44
Classifier.	0.8109	0.5412	77.83

Table 8.1: *Mean weights of the terms in the observation model evaluated on 50 frames from AAU1 (3,178 particle values) and 150 frames from Friis1 (3092 particle values). The ratio column shows the mean scaled value of Detection from AAU and Friis in the first row, the ratio between Detection and Det. conf. in the second row and the ratio between Detection and Classifier in the third row. The values in the ratio column are scaled so that the mean value of Detection is 20.00.*

Table 8.1 shows that letting $\beta = 1$, $\gamma = 1$ and $\eta = 1$ in the observation model yields a ratio between the terms in the observation model of approximately $20:16.44:77.83$. Setting the weights to $\beta = 18.8583$, $\gamma = 2.2916$ and $\eta = 0.2420$ yields a ratio of approximately $20:2:1$.

8.3 Experiment

The "Multi-Human Tracker" module is evaluated through the experiments described in Appendix A. The results of the experiments are compared with the Module Specification in Section 5.3. In the following, the requirements for the module are listed along with the results of the experiments performed on the AAU1 and Friis1 datasets.

M.2 **Multi-Human Tracker**. This module must:

a) When provided with 50% of all detections in the sequence as GT:
   1: Track people with an avg. MOTA of min. 60%.
   AAU: 67.76%.
   Friis: 32.22%.
   2: Track people with a False Positive (FP) rate less than 20%.
   AAU: 16.26%.
   Friis: 31.82%.

b) When provided with 75% of all detections in the sequence as GT:
   1: Track people with an avg. MOTA of min. 70%.
   AAU: 87.57%.
   Friis: 52.43%.
   2: Track people with an FP rate less than 15%.
   AAU: 10.51%.
   Friis: 31.22%.

c) When provided with 100% of all detections in the sequence as GT:

1: Track people with an avg. MOTA of min. 80%.
AAU: 93.82%.
Friis: 74.09%.

2: Track people with an FP rate less than 10%.
AAU: 5.20%.
Friis: 19.08%.

d) Comply with the following relation between average time consumption and the number of targets being tracked:

$$t_c < 10 \text{ for } n = 0 \quad \text{[ms]}$$
$$t_c < 250 + (n-1) \cdot 150 \text{ for } n \geq 1 \quad \text{[ms]}$$

AAU/Friis: The average time consumptions for AAU1 and Friis are listed below.

Nr. of targets	Avg. time cons.	Requirement
0.	8ms.	10ms.
1.	251ms.	250ms.
2.	325ms.	400ms.
3.	416ms.	550ms.
4.	547ms.	700ms.
5.	839ms.	850ms.
6.	1,020ms.	1,000ms.
7.	1,209ms.	1,150ms.
8.	1,081ms.	1,300ms.
9.	997ms.	1,450ms.
10.	1,100ms.	1,600ms.

8.4 Discussion

The results show that the performance of the module in terms of MOTA and FP rate fulfill the requirements when tested on AAU1. However, the performance on AUU1 is significantly higher than the performance on Friis1, where the MOTA and the FP rate do not fulfill the requirements. Inspection of the Friis1 sequence shows that it is particularly more challenging than AAU1, as people tend to clump together, making it difficult for the ”Multi-Human Tracker” to track persons individually.

The average time consumption for processing the AAU1 and Friis1 data generally fulfill the requirements. In the cases with one, six and seven targets being tracked, the average time consumption exceeds the requirements marginally, i.e. by 1ms, 20ms and 59ms, respectively.

Chapter 9

Scene Estimator

Figure 9.1 shows the contexts in which the "Scene Estimator" module is placed.

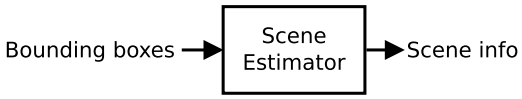


Figure 9.1: *The contexts in which the "Scene Estimator" module is placed.*

As described in the Module Specification (Section 5.3), the requirements for this module are:

M.5 **Scene Estimator**. This module must:

(a) Generate a model of the human scale factor as a function of image coordinates.

(b) The RMSE of the model must be below 0.15.

9.1 Design

Since a camera makes use of perspective projection, a person has varying pixel heights as the distance between the camera and the person changes. Like this, the "Human Detector" module has to look for people of different sizes in all areas of the image. In this work, it is assumed that people walk on a flat surface. This assumption is used to reduce the search area of the "Human Detector" by estimating a model, which expresses the average pixel height of people as a function of image coordinates. The goal for this module is to estimate such a model.

In the "Human Detector" module (described in Chapter 6), the human scale factor was described. It expresses how the "Human Detector" represents the height of persons and therefore, the model has to determine the relationship between the human scale and the image coordinates. The scale factor is defined to be:

$$s = \frac{h_p}{ht_p} \qquad [\cdot]$$

where:

h_p: Pixel height of the person. [px]

ht_p: Height of the persons in the "Human Detector" training data, see Chapter 6. [px]

In the following, the problem of deriving a model of the human scale as function of image coordinates is examined analytically. As the relationship between the pixel height of a person and

the scale is constant, the pixel height will be used in the following for simplicity. For an ideal camera, a point in the real world is projected into the image plane with the pin hole model by:[34]

$$u = -f\frac{P_x}{P_z} \quad [\cdot]$$
$$v = -f\frac{P_y}{P_z} \quad [\cdot] \quad (9.1)$$

where:

P_x, P_y, P_z: The x,y and z coordinate of a point in the camera coordinate system. [m]
u, v: The image coordinate of a point projected onto the image plane. [px]

For simplicity, the P_x coordinate is fixed at zero in the following and as a result, the image coordinate u is ignored. In Figure 9.2, a setup of a surveillance camera is illustrated in a special case, where the orientation of the camera is parallel to the ground plane. The scene is seen from the x-axis (i.e., the x-axis points out of the page) with a person standing in the y, z-plane.

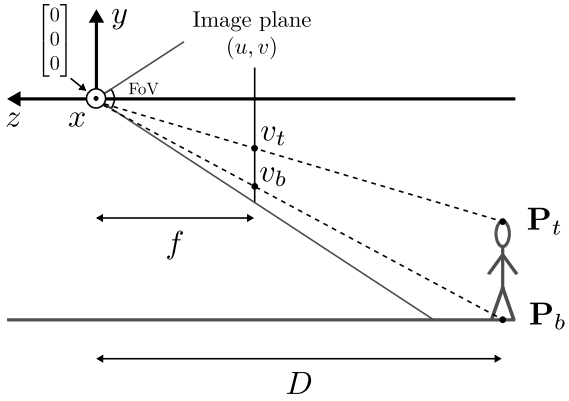


Figure 9.2: *The pinhole model illustrated in the camera coordinate system seen from the x-axis with the origin being the position of the camera. f is the focal length of the camera. D is the distance between the camera and the person. P_t and P_b are the top and lowest points of the person in camera coordinates and v_t and v_b are the top and lowest points of the person in image coordinates.*

The pixel height of the person illustrated in Figure 9.2 can be modelled as a function of the horizontal distance between the camera and the person. The top and bottom points ($\mathbf{P}_t$ and $\mathbf{P}_b$) of the person in camera coordinates are projected into the image plane using the pinhole model and the image coordinates are subtracted in Equation 9.2. In the special case where the camera is parallel to the ground plane the z-coordinate (P_z) of the two points is equal to the distance between the camera and the person ($P_{t,z} = P_{b,z} = D$).

$$\begin{aligned} h_p &= v_t - v_b && [\text{px}] \\ &= -f\frac{P_{t,y}}{D} + f\frac{P_{b,y}}{D} && [\text{px}] \\ &= -f \cdot \left(\frac{P_{t,y} - P_{b,y}}{D}\right) && [\text{px}] \quad (9.2) \end{aligned}$$

where:

v_t: Top point of the person in image coordinates.	[px]
v_b: Lowest point of the person in image coordinates.	[px]
f: Focal length.	[px]
P_t: Top point of the person in camera coordinates.	[m]
P_b: Lowest point of the person in camera coordinates.	[m]
D: Distance between the camera and the person.	[m]

Figure 9.3 shows an example of the pixel height of a person as function of the distance between the person and the camera modelled by the pinhole model. The focal length in this example is 500 pixels and the person height is 1.80m.

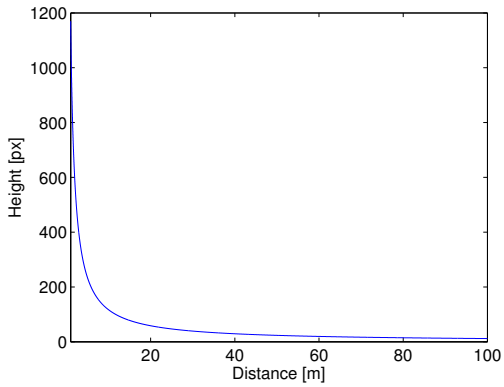


Figure 9.3: *Pixel height of a person as a function of the distance between the camera and the person.*

As Equation 9.2 and Figure 9.3 show, the relationship between the pixel height of a person and the distance from the person to the camera is nonlinear.

To investigate the relationship between pixel coordinates of a person and the pixel height (The model to be estimated in this chapter), equations expressing this relationship are derived by inspecting Figure 9.4.

An expression of the pixel height of a person as a function of the v-coordinate of the lowest point of the person (v_b) is derived from the pinhole model (Equation 9.1). For this, the camera coordinates of the top and lower point of the person ($\mathbf{P}_t$ and $\mathbf{P}_b$) are derived in Equation 9.3 and 9.4.

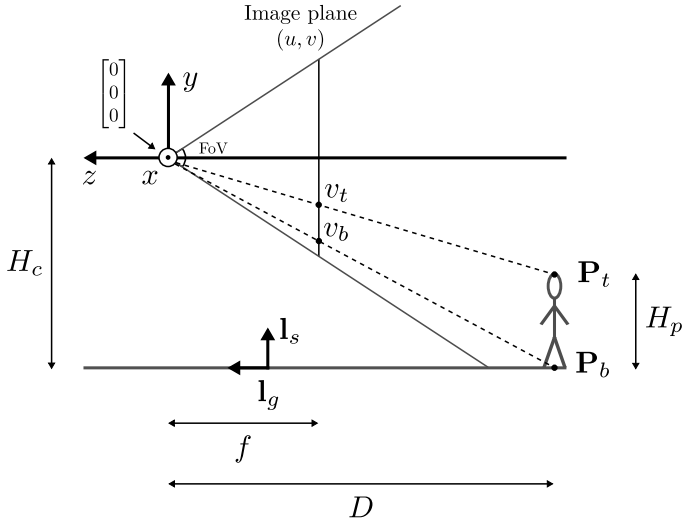


Figure 9.4: *An illustration of a simple scenario, where the orientation of the camera is parallel to the ground plane. H_c is the installation height of the camera. H_p is the height of the person. $\mathbf{P}_t$ $\mathbf{P}_b$ are the top and lowest point of the person in camera coordinates, respectively. f is the focal length. D is the distance between the camera and the person. $\mathbf{l}_g$ and $\mathbf{l}_s$ are unit vectors which are parallel and orthogonal to the ground plane, respectively. v_t and v_b are the top and lowest point of the person in image coordinates.*

$$
\begin{aligned}
\mathbf{P}_t &= (H_c - H_p) \cdot (-\mathbf{l}_s) + (D) \cdot (-\mathbf{l}_g) && \text{[m]} \\
&= (H_c - H_p) \cdot - \begin{bmatrix}0\\1\\0\end{bmatrix} + (D) \cdot - \begin{bmatrix}0\\0\\1\end{bmatrix} && \text{[m]} \\
&= \begin{bmatrix}0\\-H_c + H_p\\-D\end{bmatrix} && \text{[m]} \quad (9.3) \\
\mathbf{P}_b &= (H_c) \cdot (-\mathbf{l}_s) + (D) \cdot (-\mathbf{l}_g) && \text{[m]} \\
\mathbf{P}_b &= (H_c) \cdot - \begin{bmatrix}0\\1\\0\end{bmatrix} + (D) \cdot \begin{bmatrix}0\\0\\1\end{bmatrix} && \text{[m]} \\
&= \begin{bmatrix}0\\-H_c\\-D\end{bmatrix} && \text{[m]} \quad (9.4)
\end{aligned}
$$

where:

H_c: Installation height of the camera.	[m]
H_p: Height of the person.	[m]
$\mathbf{l}_g$: Unit vector parallel to the floor.	[·]
$\mathbf{l}_s$: Unit vector orthogonal to the floor.	[·]

The pixel height of a person as a function of the v_b-coordinate is derived in Equation 9.5.

$$
\begin{aligned}
v_b &= -f \cdot \frac{P_{b,y}}{P_{b,z}} = -f \cdot \frac{-H_c}{-D} \iff D = -f \cdot \frac{H_c}{v_b} && \text{[px]} \\
h_p &= v_t - v_b \\
&= -f \cdot \frac{P_{t,y}}{P_{t,z}} - v_b \\
&= -f \cdot \frac{-(H_c - H_p)}{-D} - v_b \\
&= -f \cdot \frac{H_c - H_p}{f \cdot \frac{H_c}{v_b}} - v_b \\
&= v_b \left(\frac{H_p - H_c}{H_c} - 1 \right) \text{ for } H_c, D > 0 && \text{[px]} \quad (9.5)
\end{aligned}
$$

where:

$P_{t,y}, P_{t,z}$: The y and z camera-coordinates of the top point of the person.	[m]
$P_{b,y}, P_{b,z}$: The y and z camera-coordinates of the lowest point of the person.	[m]
v_t: The v image-coordinate of the top point of the person.	[px]
v_b: The v image-coordinate of the lowest point of the person.	[px]

Equation 9.5 show a linear relationship between the pixel height and v_b-coordinate of the person in the case of the camera orientation being parallel with the ground plane. This is not a realistic camera configuration, therefore the camera is rotated around the x-axis to yield the general scenario illustrated in Figure 9.5.

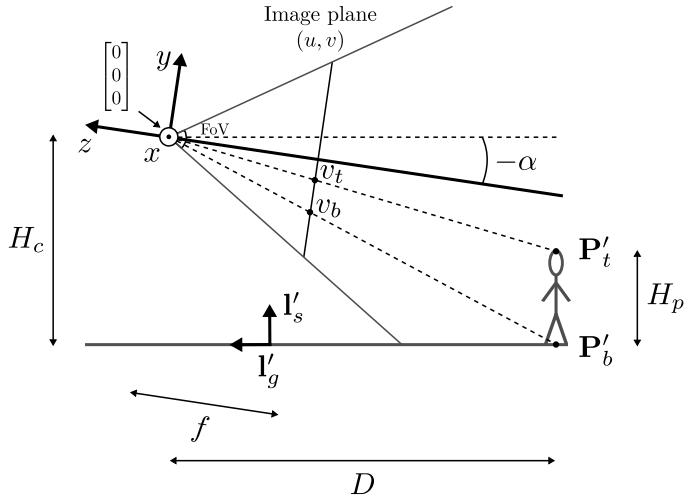


Figure 9.5: *An illustration of the general scenario, where the camera is rotated by $-\alpha$ degrees. $\mathbf{P}'_t$ and $\mathbf{P}'_b$ are the top and lowest points of the person in the rotated camera coordinate system. $\mathbf{l}'_g$ and $\mathbf{l}'_s$ are rotated unit vectors parallel and the orthogonal to the ground plane, respectively.*

From the general case illustrated in Figure 9.5 a general expression of the pixel height of a person as a function of v_b is derived. For this, the points P'_t and P'_b representing the top and lower point of the person in the general scenario are obtained by rotating the camera and the therefore also the camera coordinate system around the x-axis by α degrees. This is expressed in Equation 9.6 and Equation 9.7.

$$
\begin{aligned}
P'_t &= (H_c - H_p) \cdot (-\mathbf{l}'_s) + D \cdot (-\mathbf{l}'_g) && \text{[m]} \\
&= -(H_c - H_p) \cdot \begin{bmatrix} 0 \\ \cos\alpha \\ \sin\alpha \end{bmatrix} - D \cdot \begin{bmatrix} 0 \\ -\sin\alpha \\ \cos\alpha \end{bmatrix} && \text{[m]} \\
&= \begin{bmatrix} 0 \\ (-H_c + H_p) \cdot \cos\alpha + D \cdot \sin\alpha \\ (-H_c + H_p) \cdot \sin\alpha - D \cdot \cos\alpha \end{bmatrix} && \text{[m]} \quad (9.6) \\
P'_b &= H_c \cdot (-\mathbf{l}'_s) + D \cdot (-\mathbf{l}'_g) && \text{[m]} \\
&= -(H_c) \cdot \begin{bmatrix} 0 \\ \cos\alpha \\ \sin\alpha \end{bmatrix} - D \cdot \begin{bmatrix} 0 \\ -\sin\alpha \\ \cos\alpha \end{bmatrix} && \text{[m]} \\
&= \begin{bmatrix} 0 \\ -H_c \cdot cos\alpha + D \cdot \sin\alpha \\ -H_c \cdot sin\alpha - D \cdot cos\alpha \end{bmatrix} && \text{[m]} \quad (9.7)
\end{aligned}
$$

where:

$\mathbf{l}'_g$: Unit vector parallel to the floor in rotated camera coordinate system. [·]
$\mathbf{l}'_s$: Unit vector orthogonal to the floor in rotated camera coordinate system. [·]
α: Rotation angle of the camera. [deg]

The pixel height of a person in the general scenario is derived in Equation 9.8 by substituting 9.1 into the equation of the pixel height:

$$
\begin{aligned}
h_p &= v_t - v_b \\
&= -f \cdot \left(\frac{P'_{t,y}}{P'_{t,z}} - \frac{P'_{b,y}}{P'_{b,z}} \right) \\
&= -f \cdot \left(\frac{(-H_c + H_p) \cdot cos\alpha + D \cdot sin\alpha}{(-H_c + H_p) \cdot sin\alpha - D \cdot cos\alpha} - \frac{-H_c \cdot cos\alpha + D \cdot sin\alpha}{-H_c \cdot sin\alpha - D \cdot cos\alpha} \right) \\
&\text{for } P'_{t,z}, P'_{b,z} > 0 && \text{[px]} \quad (9.8)
\end{aligned}
$$

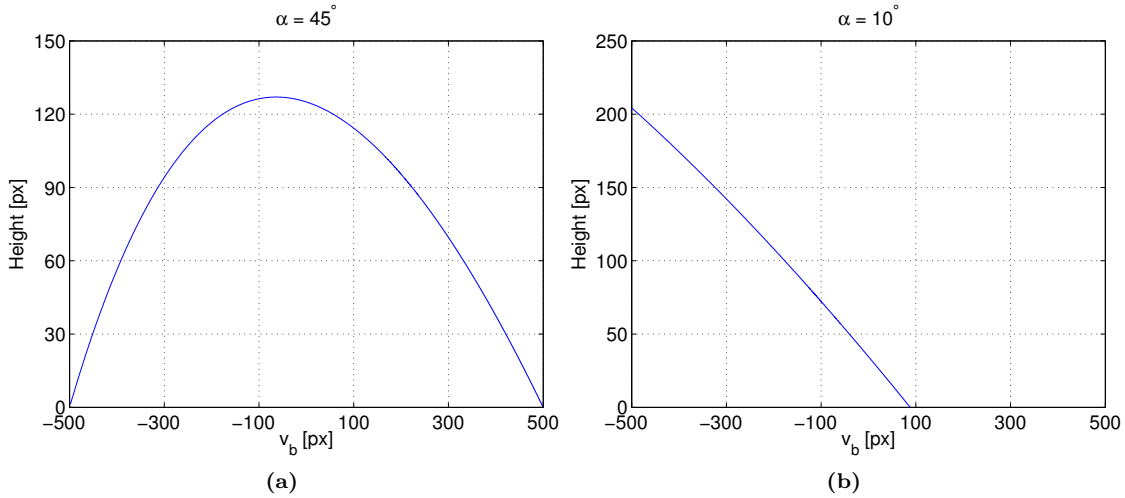
where:

$P'_{t,y}, P'_{t,z}$: The y' and z' camera-coordinates of the top point of the person. [m]
$P'_{b,y}, P'_{b,z}$: The y' and z' camera-coordinates of the lowest point of the person. [m]

Figure 9.6 shows two examples of the relationship between the pixel height and v_b in the general scenario. In both examples, $f = 500$px, $H_c = 5$m and $H_p = 2$m. In Figure 9.6a, $\alpha = 45°$ and in Figure 9.6b, $\alpha = 10°$. The second-axis is set to $[-500 : 500]$. It should be noted that no limit has been set on the FoV, and that the v_b vector is calculated by ranging D from 10cm to 5km. In practice, the FoV will be limited, and in an indoor scenario, the range of the camera to person distance will typically be much smaller. A limited FoV and a smaller distance range will truncate the curves in Figure 9.6.

Both figures show a nonlinear relationship between the pixel height of a person and v_b. However, as the angle is decreased from 45° in Figure 9.6a to 10° in Figure 9.6b, the curve is flattened. Thus, with a limited FoV, the pixel height as a function of v_b in the example, represented by the curve in Figure 9.6b can be approximated by a line.

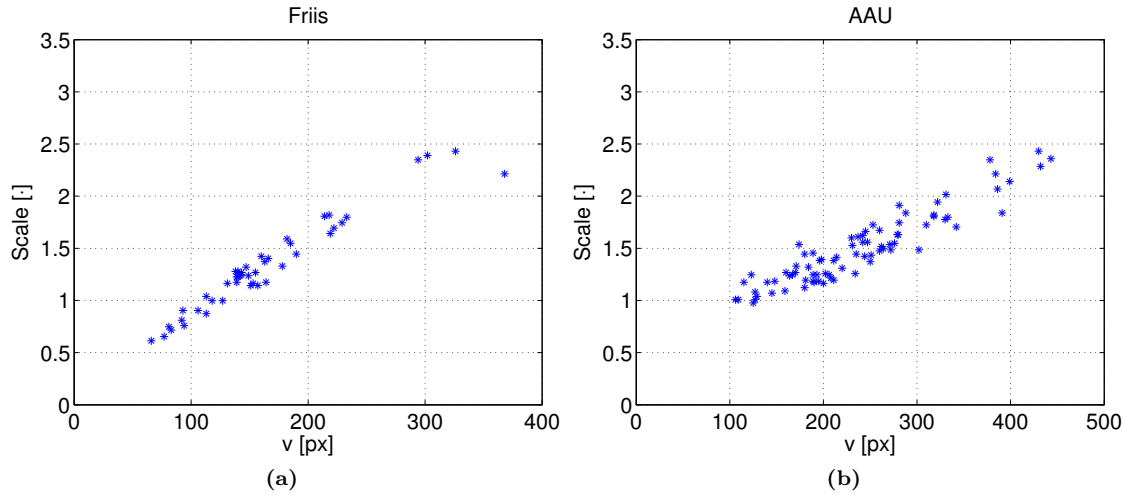
Now it is investigated if the relationship in the actual data used in this work, i.e. the data from AAU, and Friis have a linear relationship. Positions and sizes of persons in the frames are annotated manually in various frames from the datasets AAU1 and Friis1. These two new datasets



(a) (b)

Figure 9.6: *Two examples of the relationship between the pixel height of a person and v_b. In (a), the rotation angle α is set to 45° . In (b), α is set to 10°.*

are used later for verification of this module. The result of this experiment is shown in Figure 9.7. It should be noted that the axes used in the example in Figure 9.6 and in the test in Figure 9.7 are different, i.e. in Figure 9.6, the origin of the image coordinate system is the center of the image, while in Figure 9.7, the origin is the upper left corner of the image, with the positive v-direction being the opposite than that of Figure 9.6. Furthermore the human scale is viewed on the second-axis instead of the human height in pixels.



(a) (b)

Figure 9.7: *Results of the test showing the relationship between h_p and v_b performed on the Friis and the AAU dataset. (a) shows the results from the Friis data, and (b) shows the results from the AAU data. The u-axis is omitted, and thus the data is projected from 3D to 2D, which causes the points to appear more spread in (b) than (a).*

To estimate a model, an assumption is introduced. It is assumed, that the surface, on which the persons are tracked is planar. For this reason, the model to fit the data is chosen to be a polynomial, which is fitted using least squares. [43] The polynomial models of different orders are evaluated by their RMSE, which are calculated as explained by Equation 9.9. [44]

$$RMSE(d, \mathbf{c}, \mathbf{s}) = \sqrt{E[(d(\mathbf{c}) - \mathbf{s})^2]} \quad [\cdot] \quad (9.9)$$

where:

d: Model.	$[\cdot]$
$d(\mathbf{c})$: Output of the model (human scale).	$[\cdot]$
$\mathbf{c}$: Model input (image coordinates).	$[\cdot]$
$\mathbf{s}$: Human scales.	$[\cdot]$

Table 9.1 shows the $RMSE$ values of models of different orders fitted on the points shown in Figure 9.7.

Order (u)	Order (v)	$RMSE$ (AAU)	$RMSE$ (Friis)
1	1	0.1316	0.1490
2	1	0.1332	0.1469
1	2	0.1329	0.1302
2	2	0.1338	0.1275
3	3	0.1278	0.1162

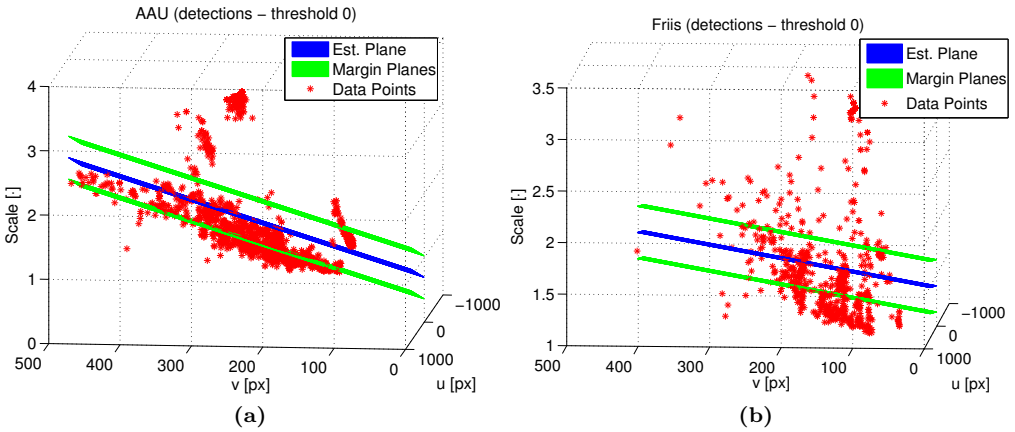
Table 9.1: *$RMSE$ of the models fitted on the AAU and the Friis data, respectively, as a result of varying u and v orders.*

The results show that both the Friis and the AAU data can be modelled accurately by a plane. Although higher model orders generally increase the precision slightly, it is chosen to model the human scale as a plane as this yields a more general model that is also computationally light. To cope with outliers caused by erroneous output from the "Human Detector" or the "Multi-Human tracker" module, the plane is estimated in two passes. In the first pass, outliers are removed, and in the second, the plane is estimated. This is done to estimate a plane that models the actual human scale more accurately.

Outliers are removed by estimating the best fitting plane from all elements in the data. An upper and a lower margin plane are then defined by offsetting the estimated plane positively and negatively by a scale margin along the z-axis. For the "Human Detector", the SVM threshold can be increased, which results in the module making fewer but increasingly reliable detections.

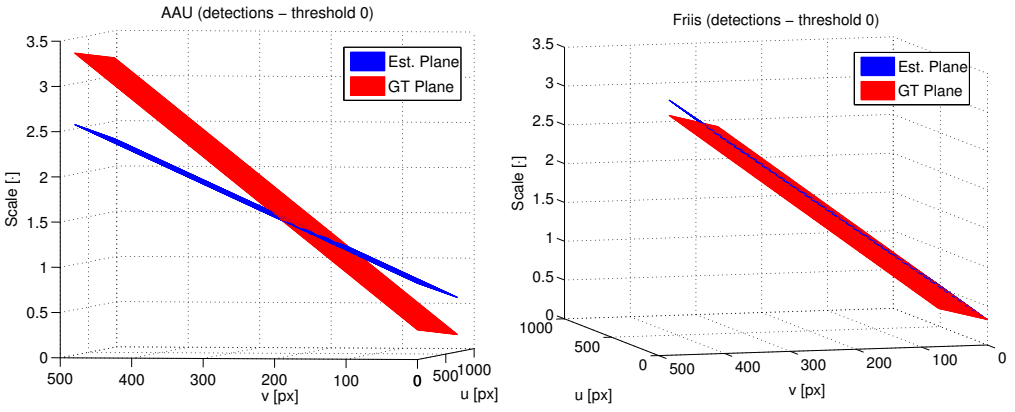
In the Figures 9.8, 9.10 and 9.12 the output from the "Human Detector" with estimated planes and the margin planes are shown at SVM thresholds 0,1 and 2, respectively.

In the Figures 9.9, 9.11 and 9.13 the planes estimated on detector data at different SVM thresholds are compared to the ground truth planes. The ground truth planes are estimated on the manually annotated data from Figure 9.7.



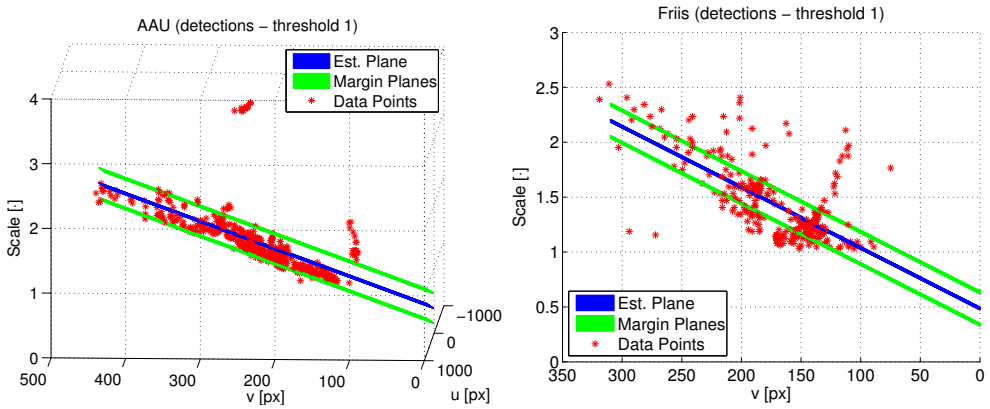
(a) (b)

Figure 9.8: *Output from the "Human Detector" with an SVM threshold of 0. (a) shows the output from the AAU dataset. (b) shows the output from the Friis dataset.*



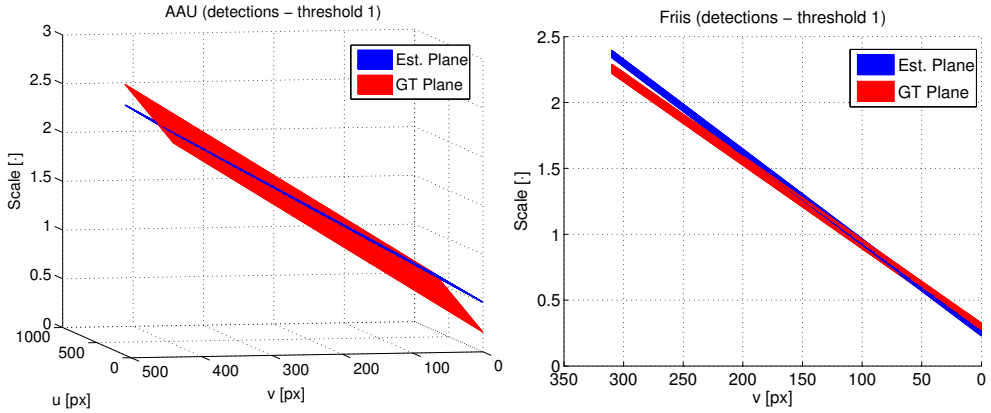
(a) (b)

Figure 9.9: *Comparison between estimated planes and ground truth planes on AAU dataset (a) and Friis dataset (b) with SVM threshold 0.*



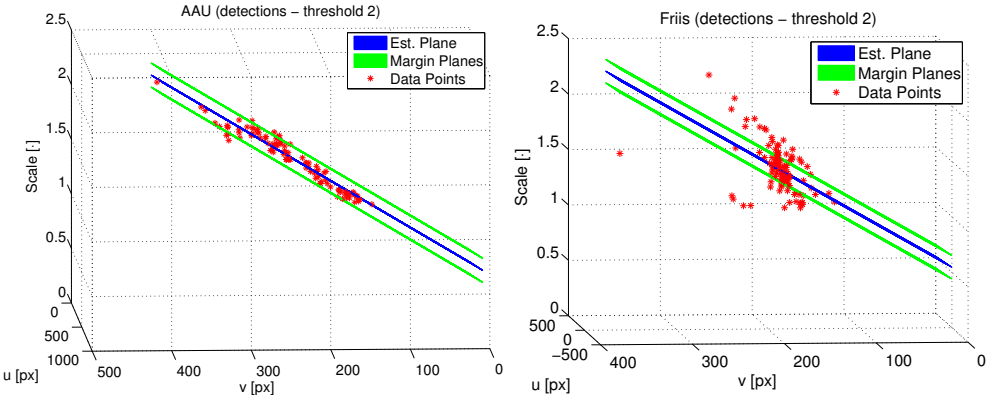
(a) (b)

Figure 9.10: *Output from the "Human Detector" with an SVM threshold of 1. (a) shows the output from the AAU dataset. (b) shows the output from the Friis dataset.*



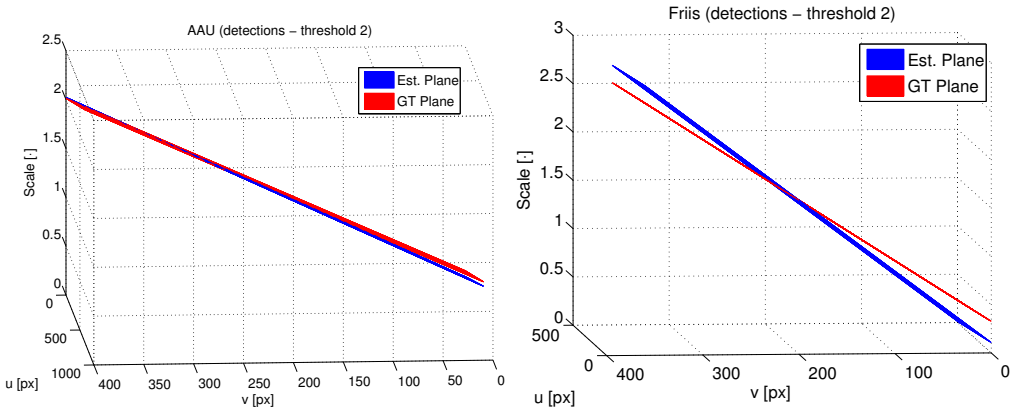
(a) **(b)**

Figure 9.11: *Comparison between estimated planes and ground truth planes on AAU dataset (a) and Friis dataset (b) with SVM threshold 1.*



(a) **(b)**

Figure 9.12: *Output from the "Human Detector" with an SVM threshold of 2.(a) shows the output from the AAU dataset. (b) shows the output from the Friis dataset.*

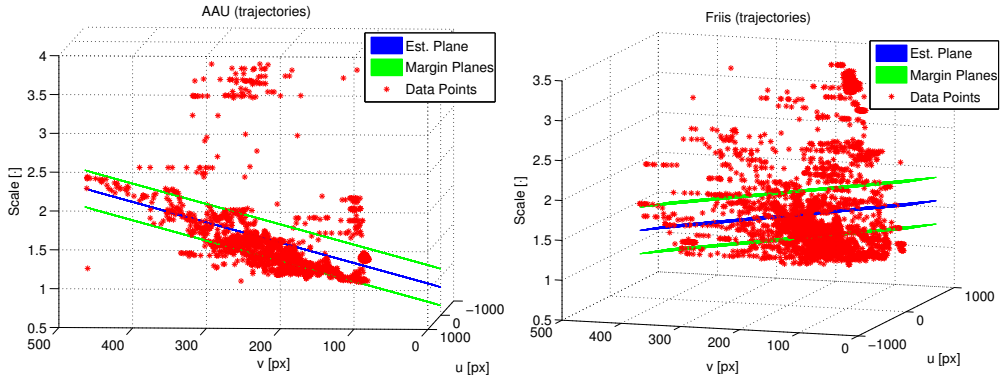


(a) **(b)**

Figure 9.13: *Comparison between estimated planes and ground truth planes on AAU dataset (a) and Friis dataset (b) with SVM threshold 2.*

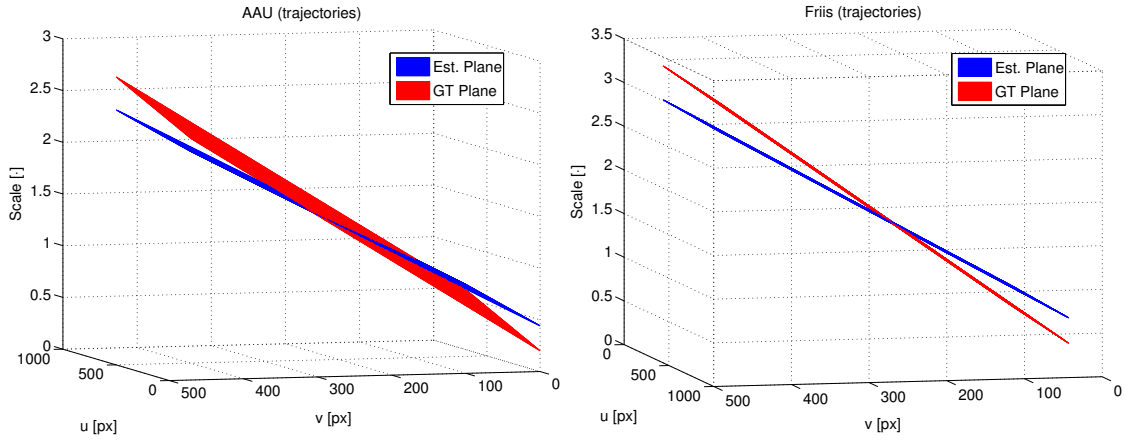
Figure 9.12 shows that an SVM threshold of 2 causes the "Human Detector" to make detections with a low error. The planes are similar to the ground truth planes estimated from the data points shown in Figure 9.7.

To assess the output from the "Multi-Human Tracker" module can be used for scene estimation, the output from "Multi-Human Tracker" is plotted in Figure 9.14 along with the estimated plane. In Figure 9.15 the estimated plane is compared to the ground truth plane.



(a) **(b)**

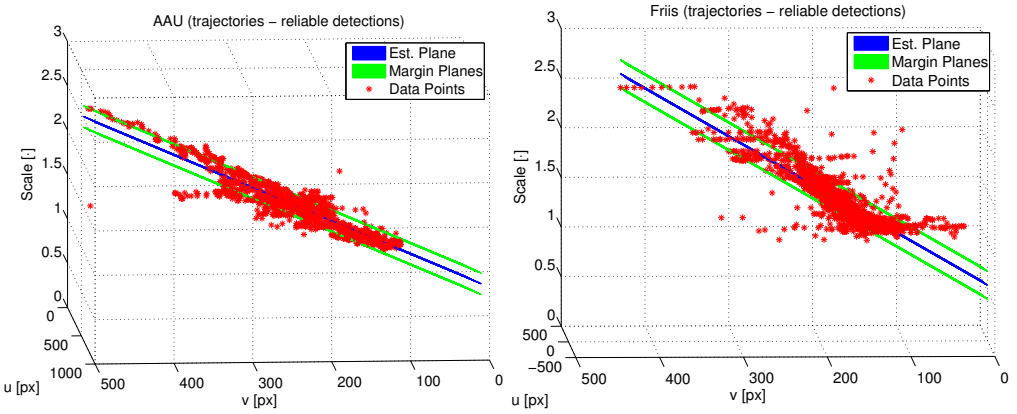
Figure 9.14: *Output from the "Multi-Human Tracker" module.(a) shows the output from the AAU dataset. (b) shows the output from the Friis dataset.*



(a) **(b)**

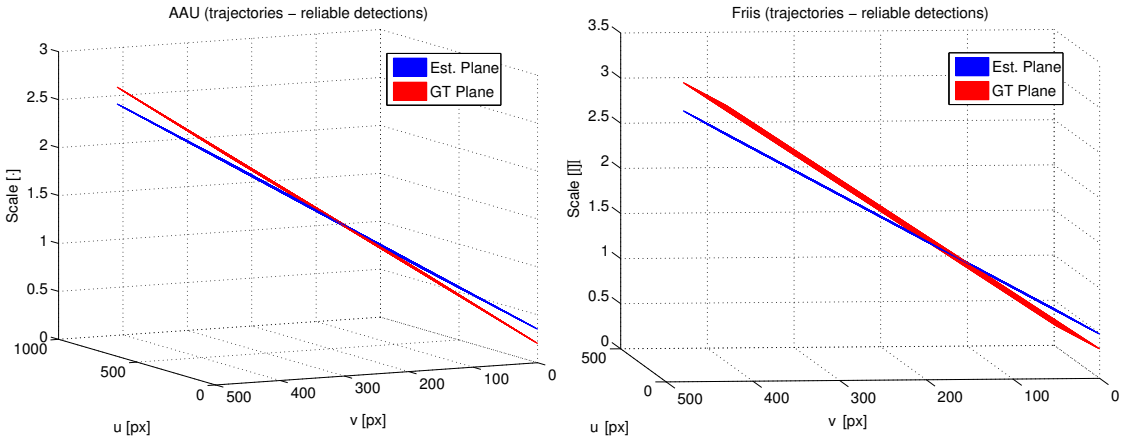
Figure 9.15: *Comparison between estimated planes and ground truth planes on AAU dataset (a) and Friis dataset (b) from the "Multi-Human Tracker" module.*

Figure 9.15 shows that the output from "Multi-Human Tracker" contains too much noise to be used for scene estimation. Unfortunately, although increasing the SVM threshold will increase the reliability of the human detections, it will also decrease the number of detections, thus decreasing the performance of the tracker. Therefore, it can be concluded that the output from the "Multi-Human Tracker" alone cannot be used for estimating the scene. Providing the "Multi-Human Tracker" with detections, where the "Human Detector" module uses a plane estimated previously from reliable detections yields the output shown in Figure 9.16. The comparison of the estimated planes to the ground truth planes are done in Figure 9.17.



(a) **(b)**

Figure 9.16: *Output from the "Multi-Human Tracker" module, with the "Human Detector" using a plane previously estimated from reliable detections.(a) shows the output from the AAU dataset. (b) shows the output from the Friis dataset.*



(a) **(b)**

Figure 9.17: *Comparison between estimated planes and ground truth planes on AAU dataset (a) and Friis dataset (b) from the "Multi-Human Tracker" using scene information.*

Figure 9.17 shows that providing the "Multi-Human Tracker" module with more reliable detections from the human detector yields more accurate results than processing the "Multi-Human Tracker" output alone. However, by comparing the similarity between the generated planes and the ground truth planes in Figure 9.17 and in Figure 9.13, it can be concluded that the most precise planes are estimated using the output from the "Human Detector" module alone. Furthermore, this way of combining the output from the "Human Detector" module and the "Multi-Human Tracker" module will require

As described earlier, the outliers of the "Human Detector" data are filtered out using two scale margin planes. The upper and a lower margin plane are defined by offsetting the estimated plane positively and negatively by a scale margin along the z-axis. The size of the offset for the margin planes has been found experimentally by comparing the final estimated plane with the GT plane at different sizes of offsets. The offset is defined to depend on the standard deviation of the data. The comparison is done by finding the maximum distance between the estimated plane and the GT plane. Figure 9.18 shows the results.

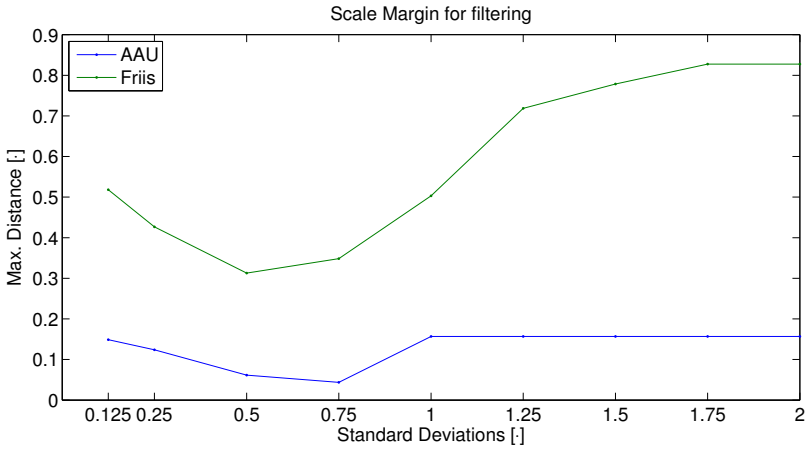


Figure 9.18: *Scale margin for filtering the "Human Detector" data. The Y-axis shows the distance between the estimated plane and the GT plane at different number of standard deviations.*

The results show that a scale margin of a half standard deviation of the data yields the most efficient filtering in terms of max. distance between the estimated plane and the GT plane. The scale margin for filtering differs from the scale margin that is to be incorporated in the model.

During runtime in the "Processing" subsystem when using the scene model estimated in this module, another margin of the scale is applied to the model. This is caused people having different sizes and thus yielding different pixel size in the same area of the image. The scale margin yielding the lowest error in the "Human Detector" is found through an experiment, where the generated planes are used for detection. The scale margin is varied from 0.1 to 1.0, and the SVM threshold is set to 0. The results are shown in Figure 9.19.

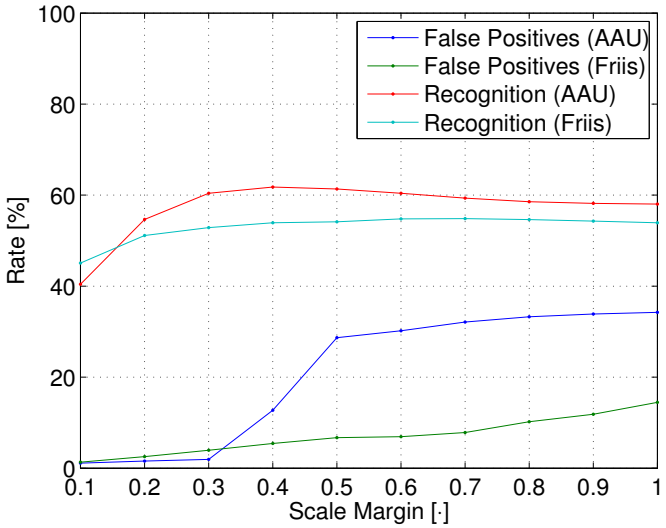


Figure 9.19: *Performance of the "Human Detector" in terms of Recognition Rate and False Positives as a result of varying scale margins.*

The figure shows that a scale margin of 0.3 produces a low number of false positives and a high recognition rate for both the AAU dataset and the Friis dataset.

9.2 Experiment

As shown in Table 9.1 on page 66, the GT containing the image coordinates and size of persons generated manually from AAU1 and Friis1 can be modelled by planes with RMSEs of 0.1316 and 0.1490, respectively. This module has introduced the following assumptions:

- The surface, on which persons are tracked must be planar.
- It must be possible to model the human scale as a function of image coordinates with a low RMSE. This depends on parameters that influences on how much of the scene that is visible for the camera, including:
  - FOV of the camera.
  - Focal length of the camera.
  - Installation height of the camera relative to the floor plane.
  - Camera angle relative to the floor plane. Vertical and horizontal camera angles make assumptions on FOV and focal length negligible. The installation height must then be above zero.

It was shown that using detections from the Human Detector module with an SVM threshold of 2 yielded the most precise model. Furthermore, it was shown that the trajectories from the "Multi-Human Tracker" module contain too much noise to be used for generating a model.

Chapter 10

Visualization

Figure 10.1 shows the context in which the "Visualization" module is placed.

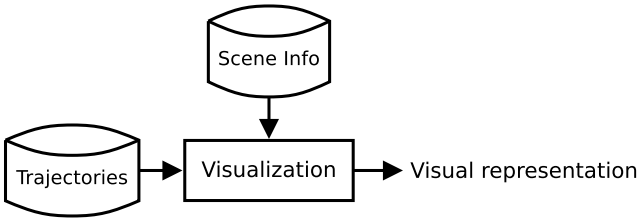


Figure 10.1: *The context in which the "Visualization" module is placed. The module gets a data structure containing trajectories as input, generates statistics and represents them visually.*

As described in the Module Specification (Section 5.3), the requirements for this module are:

M.4 **Visualization**. This module must:

1. Present heat maps of the following statistics, described in Section 2.4:
   a) Person Position with a mean error less than 0.1.
   b) Speed with a SMAPE less than 15%.
   c) Dwell Analysis with a mean error less than 0.1.
2. Present vector plots of the following two statistics, described in Section 2.4:
   1. Mean Directions;
      1. with a mean angle error less than 10°.
      2. with a magnitude SMAPE less than 15%.
   a) Top-Four Directions with a SMAPE less than 15%.
3. Present a Number of Persons plot (described in Section 2.4) with a mean error less than 1.5.

10.1 Module Overview

The "Visualization" module receives estimated trajectories from the "Multi-Human Tracker" module of the posistion and size of people over time. According to the requirements, this module must generate statistics on the movement of people and visualize these. The module has been divided into three steps: First, the trajectories are prepared in a preprocessing step. Then, the

actual statistics are calculated and lastly, the statistics are visualized. The three steps are designed in the following three sections.

10.2 Preprocessing

In the preprocessing step of this module, the raw trajectories from the "Multi-Human Tracker" module are prepared to be used for statistics generations.

The output from the "Multi-Human Tracker" module is a number of human trajectories, expressing where the tracker has estimated each human to have been over time. The trajectories are represented as a sequence of bounding boxes that denote the estimated position and size of humans in each frame. The statistics to be calculated in this module needs the coordinates of the feet of the people. Therefore, the bounding boxes must be converted to feet coordinates of the targets. The bounding boxes are given by a coordinate of the upper left corner (x_b, y_b) and the width and height (w_b, h_b). Therefore, the bottom-center coordinate (x_{bc}, y_{bc}) of a bounding box is given by:

$$x_{bc} = x_b + \frac{w_b}{2} \quad [\text{px}]$$
$$y_{bc} = y_b + h_b \quad [\text{px}]$$

where:

x_b, y_b: Coordinates of upper left corner of a bounding box. [px]
x_{bc}, y_{bc}: Coordinates of buttom center of a bounding box. [px]
w_b, h_b: Width and height of the boinding box, respectively. [px]

The bottom center of the bounding box is found next. This coordinate is however not the coordinates of the feet of the humans. The bounding box includes a margin, which was introduced in the "Human Detector" module. Recall that the margin between humans and the 64×128 windows of the detector is 16 pixels in top and bottom of the bounding box. This means that the ratio between the bottom margin and bounding box height is 0.125. To calculate the feet coordinates, the margin is subtracted from the bottom-center coordinate of the boundary box:

$$\begin{aligned} x_f &= x_{bc} && [\text{px}] \\ &= x_b + \frac{w_b}{2} && [\text{px}] \\ y_f &= y_{bc} - h_b \cdot 0.125 && [\text{px}] \\ &= y_b + h_b - h_b \cdot 0.125 && [\text{px}] \end{aligned}$$

where:

x_f, y_f: Coordinates of the feet. [px]

This is how every bounding box in the raw output from the "Multi-Human Tracker" module is converted into feet coordinates of the humans. The resulting sequence of feet coordinates for each target is for now on referred to as feet trajectories. In Figure 10.2, one short trajectory is illustrated as example for the conversion of the trajectories from the "Multi-Human Tracker" to feet trajectories.

It is seen in Figure 10.2b that the feet trajectories includes noise introduced by the "Multi-Human Tracker". This will affect the statistics generation in the following sections. Therefore, the trajectories are filtered with a smoothing filter. The filter calculates the average of a finite number of neighbors with center in every data point and is implemented by convoluting a kernel with the sequence. The two coordinates, x_f and y_f in a feet trajectory are filtered individually. The kernel elements are the inverse of the size of the kernel:

$$\mathbf{k} = [1/o \;\; 1/o \;\; \ldots \;\; 1/o] \quad [\cdot]$$

(a) **(b)**

Figure 10.2: *(a): The raw ouput from the "Multi-Human Tracker" module is a sequence of bounding boxes for each human that it tracks. (b): The feet trajectories converted from the boundin boxes.*

where:

k:	The average smoothing kernel with o elements.	$[\cdot]$
o:	The size of the kernel.	$[\cdot]$

When using a smoothing kernel on a finite sequence, the output sequence will be $\lfloor o/2 \rfloor \cdot 2$ long. This is caused by the kernel extending out of the sequence in the beginning and the end of the sequence. Therefore, the output sequence is not defined there.

In Figure 10.3, three subfigures illustrate the effect when using three different kernel sizes; 5, 11, 21. The kernel with the size of 21 shown in Figure 10.3c yields the straightest line but still retain the overall path of the trajectory. However, the output will be 20 data points shorter, which is a big a loss for the small sequences. In Figure 10.3a, the kernel size is 5. Here it is seen the that noise has a great impact on the filtered trajectories. Therefore, compromise is done by using a kernel size of 11.

10.3 Generate Statistics

In this section, the calulation of the statistics defined in Section 2.4, "Movement Statistics" is described.

10.3.1 Person Position

The calculation of the position heat map is described in this section. In Section 2.4 (Movement Statistics), it was stated that it is relevant for Friis to monitor how many people that walk through certain areas in the image. This is what the Person Position heat maps can show.

The heat maps are calculated by counting the number of times that a different trajectory is passing through each bin. It is calculated by running through all points in each trajectory. For each point, the bin it falls into is incremented. The bin ID is recorded and if another point from the same trajectory falls into the same bin later the bin count is not incremented again. This is done to avoid subsequent trajectory points that normally lie close to be counted multiple times in the same bin, and thereby give a false impression number of persons passing through that bin.

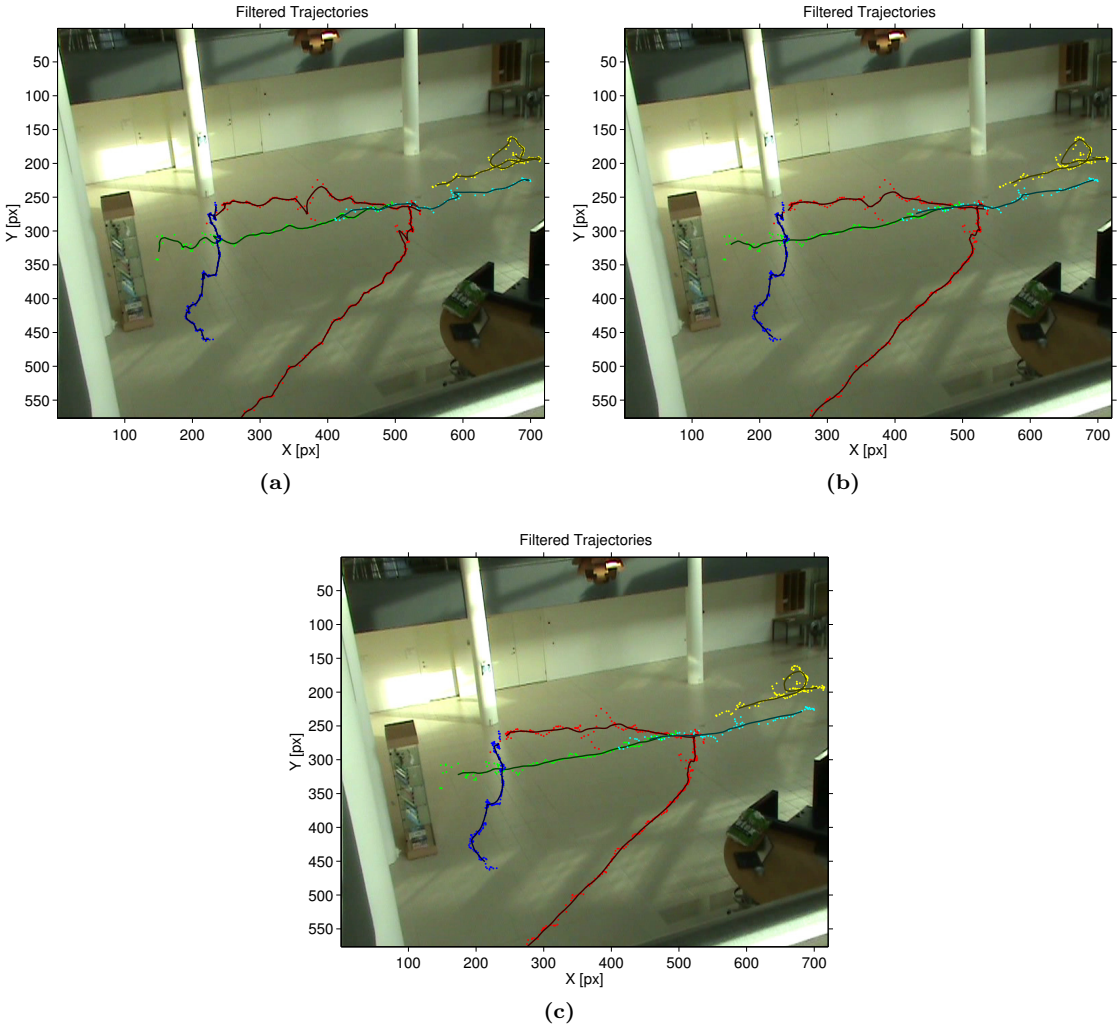


Figure 10.3: *Unfiltered feet trajectories are plotted as points, while filtered feet trajectories are plotted as lines. (a): Kernel size 5. (b): Kernel size 11. (c): Kernel size 21.*

10.3.2 Speed

In the following, the calculation of the speed is described. A consistent speed is desired, i.e. equal scene distances seen at different areas in the frames must yield the same speed. Using the image coordinates to calculate the speed, will result in an inconsistent speed, as the pixel difference is larger for persons walking close to the camera. The average speed in each heat map bin is calculated relative to other bins. Thus, a consistent speed is desired rather than an exact speed.

Using the pinhole model described in the "Scene Estimator" module (Chapter 9), the camera coordinates of a person is derived:

$$P_x = -P_z \cdot \frac{u}{f} \quad [\text{m}] \quad (10.1)$$

$$P_y = -P_z \cdot \frac{v}{f} \quad [\text{m}] \quad (10.2)$$

$$P_z = -D \quad [\text{m}] \quad (10.3)$$

where:

u, v: Image coordinates of the person. [px]
f: Focal length. [px]
D: Depth between the camera and the person. [m]

Likewise, the pinhole model is used to derive an expression for the pixel height of a person as a function of camera coordinates as expressed in Equation 10.4.

$$h_p = f \cdot \frac{P_{y_t} - P_{y_b}}{D} = f \cdot \frac{H_p}{D} \quad \text{[px]} \quad (10.4)$$

where:

H_p: Height of the person. [m]
P_{y_t}, P_{y_b}: Top and lowest scene y-coordinate of a person, respectively. [m]

Calculating the pixel height using Equation 10.4 requires a known focal length, a known height of the person and a known depth between the camera and the person. Assuming a known focal length and a known height of the person, the depth between the camera and the person can be calculated using Equation 10.5.

$$f \cdot \frac{H_p}{D} - h_p = 0 \iff D = H_p \cdot \frac{f}{h_p} \quad [\cdot] \quad (10.5)$$

where:

h_p: Pixel height of the person. [px]

Combining the above equations, i.e. Equation 10.1, 10.2, 10.3, 10.4 and 10.5, expressions for the scene coordinates of a person as a function of image coordinates are derived in Equation 10.6, 10.7 and 10.8.

$$P_x = -P_z \cdot \frac{u}{f} = H_p \cdot \frac{u}{h_p} \quad \text{[m]} \quad (10.6)$$

$$P_y = -P_z \cdot \frac{v}{f} = H_p \cdot \frac{v}{h_p} \quad \text{[m]} \quad (10.7)$$

$$P_z = -H_p \cdot \frac{f}{h_p} \quad \text{[m]} \quad (10.8)$$

An exact speed is not desired, but rather a speed that is consistent throughout the image and can be used to compare heat map bins. Therefore, constant scale factors of the speed are omitted. It is seen that the height of the person, H_p is present in the expressions for both P_x, P_y and P_z, and thus becomes a constant scale factor, which therefore is omitted. Also, instead of the pixel height of the person, h_p the model for the human scale factor, s as a function of image coordinates, estimated by the Scene Estimator module is used, as the relationship between h_p and s is constant. This yields the scaled coordinates expressed by Equation 10.9, 10.10 and 10.11.

$$P_{s_x} \equiv \frac{u}{s} \quad [\cdot] \quad (10.9)$$

$$P_{s_y} \equiv \frac{v}{s} \quad [\cdot] \quad (10.10)$$

$$P_{s_z} \equiv -\frac{f}{s} \quad [\cdot] \quad (10.11)$$

where:

s: Human scale factor. $[\cdot]$

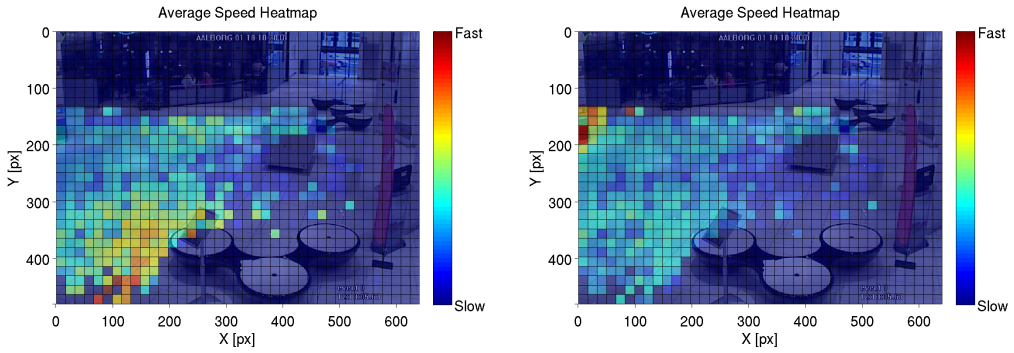
Assuming a constant framerate, the speed is defined as the distance between the scaled camera coordinates in frame i and $i-1$, denoted P_{s_i} and $P_{s_{i-1}}$ per frame as expressed in Equation 10.12.

$$speed \equiv ||P_{s_i} - P_{s_{i-1}}|| \quad [\cdot] \quad (10.12)$$

where:

$P_{s_i}, P_{s_{i-1}}$: Scaled scene coordinates of a person in frame i and $i-1$, respectively. $[\cdot]$

Figure 10.4 shows a visualization of the calculated speed. In Figure 10.4a, the "raw" speed is calculated as the pixel difference of person coordinates between the current and the previous frame. In Figure 10.4b, the speed is calculated as described in Equation 10.12.



(a) **(b)**

Figure 10.4: *Visualization of the calculated Speed. Figure (a) shows the speed calculated as the pixel difference between frames. Figure (b) shows the speed calculated as in Equation 10.12.*

It is seen that for the "raw" speed in Figure 10.4a, the closer to the image border the persons are walking, the higher their speed is. In Figure 10.4b, the speed is more uniformly distributed.

Calculating the speed assumes either a known or at least rough estimate of the focal length of the camera. An alternative to this is to find the homography between the image plane and the ground plane and then use the ground plane coordinates to calculate the speed. However, this requires that the user manually provide the system with at least four points on the ground plane and their corresponding points on the image plane. [34] As this work is aimed at generating the statistics automatically, a known or a rough estimate of the focal length is assumed for calculating the speed.

A focal length that is common for Friis and AAU is estimated. For this, equal scene distances at different image locations were found by exploiting the tile structures of the floors (see Figure 2.12). The focal lengths were then estimated by minizing Equation 10.12 iteratively for the focal lengths. A focal length of 1500px yields approximately the best common focal length for AAU and Friis. This focal length would then be used in the case, where video sequences from other cameras was to be processed by the system.

10.3.3 Dwell Analysis

The estimated Speed heat map is used for generating the Dwell Analysis heat map. The Dwell Analysis heat map express how many persons inside each bin that have slowed down to move

at a speed lower than a defined threshold. A high threshold yields a Dwell Analysis heat map that resembles the Position heat map, while lower threshold values yields the desired heat map that shows where people tend to dwell. A threshold of zero is likely to result in a heat map with all-zero bins as the probabilistic nature of the particle filtering framework used for tracking causes the estimated target speed to practically never be zero. This threshold is set by the user, and has been set to 0.5 for this work as an example.

10.3.4 Mean Direction

In this section, the calculation of the Mean Direction statistics is described. The Mean Direction is used to show if there is a tendency of people moving in spedicif directions at certain areas of the scene.

A mean direction is calculated for each bin in the image grid. It is done by running through the trajectory points of all trajectories. For each point, the vector between it and the next in the sequence is calculated. The vector is normalized and accumulated in the bin that it belongs to, which is determined from the position of the first of the two subsequent points. When this has been done for all trajectories, the accumulated vectors for each bin are normalized with the number of vectors accumulated in that bin.

By calculating the mean directions with this method, the length of the resulting vector is an expression for how consistent the direction of people is for a certain bin. This means that if a bin is exposed to equal amounts of people moving opposite directions, the vector will be short.

10.3.5 Top-Four Directions

In the previous section, the calculation of the Mean Direction was described. That statistic is able to show a resulting direction of all persons passing through a bin. If the people passing through a bin are walking in two perpendicular directions, the resulting mean vector will not indicate either of the directions, but rather an average between them. This is where the Top-Four direction plot can be useful, since it shows the four most used directions. The calculation of the Top-Four direction histogram is described in the following.

The Top-Four Direction histogram is three dimensional histogram. The first two dimensions are the normal bins, which divide the frame into a grid. For each of these spatial bins, an eight bin angle histogram is assigned. The bins of the angle histogram are spread over 360°. The calculation of the Top-Four Direction histogram is done by first calculating the angle for each point in the trajectories. The angle is calculated between the current point and the next in the sequence. The spatial bin that the first point belongs to is found and the angle histogram is updated with the calculated angle. Only the four bins with the highest counts are visualised, hence the name Top-Four Direction.

10.3.6 Number of Persons

This statistic expresses how many persons that are inside the FOV of the camera. How it is calculated is described in the following. All the trajectories are investigated for which frames they they have points in. The number of active trajectories in each frame is counted. This count is the number of persons that the ”Processing” subsystem estimate to be inside the FOV at a given time.

In order to emphasize how the tendencies of number of persons in the scene changes over a longer time period, a smoothing average kernel is used. The kernel is the same used for smoothing the trajectories. However, the kernel used for this smoothing is 1,000 elements long, meaning it calculates an average of the number of persons in the surrounding 1,000 frames at each frame. The size of the kernel corresponds to approximately 2-3 minutes of video at 5-8 FPS, which is what the Friis videos are recorded at. The length of the kernel has been set through inspection.

10.4 Visualization

In this section, the visualization of the statistics is described. Four different types of plots are necessary, since four different types of statistics have been described in the previous sections.

For the Person Position, Speed and Dwell Analysis histograms, the image has been divided into a grid and a value has been assigned to every bin. Such statistics can be visualized with heat maps, where the values in each bin is represented with a color. The color scale used give high values a red color, values in the middle a green color and the lowest values are given a blue color. The colored grid is drawn on an empty frame from the sequence, which the statistics are generated from. This makes the user able to see, where a certain high or low value is in the image and relate it to the scene.

The Mean Direction statistic is represented by vector for each bin in the image grid. This statistic is visualized with a vector field plot: An arrow for each bin is drawn on an empty frame.

The Top-Four Directions statistic contains an eight bin angle histogram for each of the spatial bins. They are visualized like the Mean Direction with a vector field plot but with four arrows instead of one. They point in the direction of the center of the angle bins with highest counts. The length express how many points that fall into the angle bin.

The Number of Persons statistics includes two different statistics: the number of person in every frame, and an average with a 1,000 frame window. The two statistics are shown in the same plot, where the Number of Persons is a bar plot with a curve of the average plotted upon.

10.5 Experiment

In this section, the experiments that have been performed to evaluate this module are described. The experiments are performed to investigate, whether the module fulfills the requirements specified in Section 5.3 in the Program Design. The requirements are listed in the following along with the result of the experiment performed on AAU1 and Friis1.

M.4 **Visualization**. This module must:

1. Present heat maps of the following statistics, described in Section 2.4:
   a) Person Position with a mean error less than 0.1.
   AAU: 0.1550.
   Friis: 0.4217.
   b) Speed with a SMAPE less than 15%.
   AAU: 9.25%.
   Friis: 17%.
   c) Dwell Analysis with a mean error less than 0.1.
   AAU: 0.0217.
   Friis: 0.1417.
2. Present vector plots of the following two statistics, described in Section 2.4:
   a) Mean Directions;
      1. with a mean angle error less than 10°.
      AAU: 11.57°.
      Friis: 15.91°.
      2. with a magnitude SMAPE less than 15%.
      AAU: 9.62%.
      Friis: 18.96%.
   b) Top-Four Directions with a SMAPE less than 15%.
   AAU: 3.69%.
   Friis: 10.79%.
3. Present a Number of Persons plot (described in Section 2.4) with a mean error less than 1.5.

AAU: 0.6520.
Friis: 1.0030.

The estimated statistics and the corresponding GT are plotted and compared qualitatively in Appendix E.

10.6 Discussion

The results show that the requirements M.4.1.a, M.4.2.a.1 are not fulfilled for AAU and Friis, while requirement M.4.1.c is fulfilled for AAU but not for Friis. The qualitative comparison of the estimated plots and the GT plot in Appendix E shows that person movement in the top right corner in the image in Friis is not captured by the system due too the persons being too far away and therefore being too small in the image, thus contributing significantly to errors in the statistics.

For more representative statistics, a larger amount of data is necessary. As this module is evaluated quantitavely, the data had to have associated GT, which has limited the amount of data for module evaluation because of the time cost of generating GT. However, in the Acceptance Test (Chapter 12), besides evaluating the system quantitatively, the system is evaluated qualitatively on large datasets, i.e. AAU4, AAU5 and Friis4 (consisting of 8134, 7534 and 20936 frames, respectively).

The module introduces the following assumptions, both used for calculating the speed:

- The focal length is either known or a rough estimate is available.
- The framerate is constant.

Chapter 11

Subsystem Test

In this chapter, the subsystem test is described. The test is performed in accordance with the subsystem specification described in Section 5.4 in the Program Design chapter. Separate tests are performed for the "Preprocessing" and the "Processing" subsystem. The subsystem test is performed on the AAU3 and the Friis3 dataset.

11.1 Preprocessing

In the following, the requirement for the "Preprocessing" subsystem is written along with the result of the test performed on AAU3 and Friis3.

S.1 **Preprocessing**. This subsystem must:

a) Generate a model that estimates the human scale as a function of image coordinates. The largest distance between the model and the GT model must be less than one standard deviation of the data.
AAU: Largest distance: 0.1113. Standard deviation: 0.2262.
Friis: Largest distance: 0.1844. Standard deviation: 0.2347.

For both AAU and Friis, the largest distance between the estimated model and the GT model is less than one standard deviation of the data. Thus, the "Preprocessing" subsystem fulfills the requirement.

11.2 Processing

The requirements for the "Processing" subsystem are listed in the following along with the results of the test performed on AAU3 and Friis3.

S.2 **Processing**. This subsystem must:

a) Track people with an avg. MOTA of min. 50%.
AAU: 76.13%.
Friis: 25.39%.

b) Track people with an FP rate less than 15%.
AAU: 12.81%.
Friis: 25.83%.

d) Comply with the following relation between average time consumption and the number of targets being tracked:

$$t_c < 1000 + n \cdot 200 \text{ for } n \geq 0 \quad \text{[ms]}$$

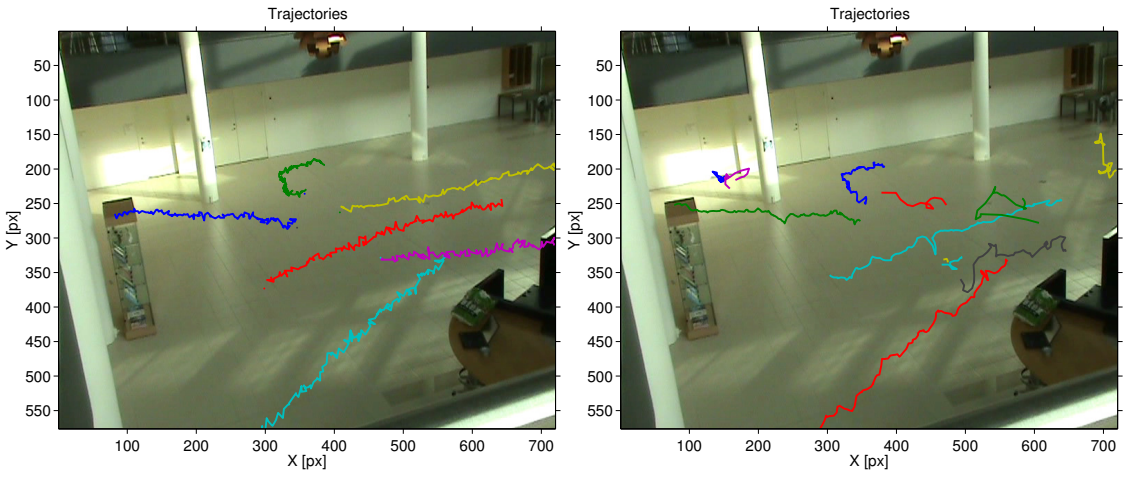
AAU: The average time consumptions for AAU3 are listed in the table below.

Nr. of targets	Avg. time cons.	Requirement
0.	823ms.	1,000ms.
3.	872ms.	1,600ms.
4.	1,816ms.	1,800ms.
5.	1,996ms.	2,000ms.
6.	2,161ms.	2,200ms.
7.	2,422ms.	2,400ms.
8.	2,539ms.	2,600ms.

Friis: The average time consumptions for Friis3 are listed in the table below.

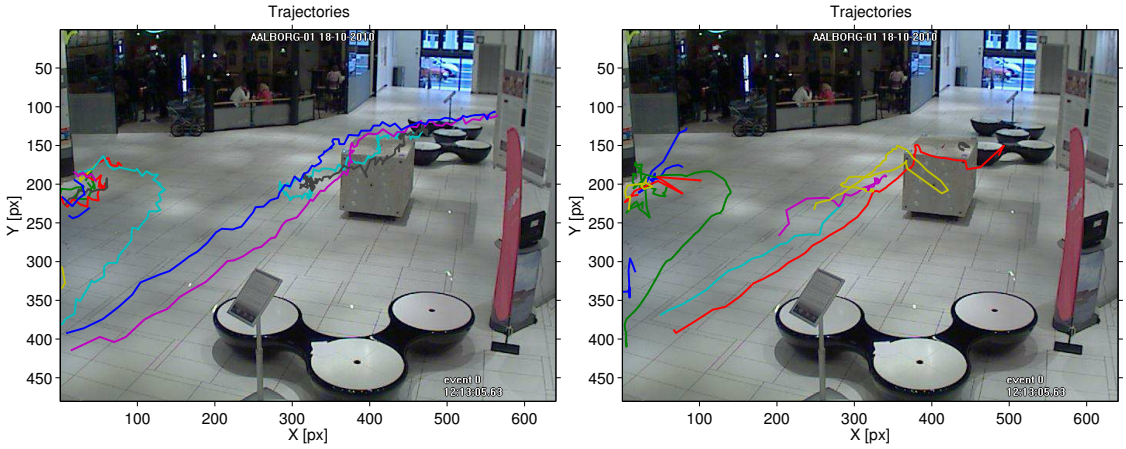
Nr. of targets	Avg. time cons.	Requirement
0.	562ms.	1,000ms.
1.	737ms.	1,200ms.
2.	1,001ms.	1,400ms.
3.	1,383ms.	1,600ms.
4.	1,666ms.	1,800ms.
6.	2,182ms.	2,200ms.
7.	2,134ms.	2,400ms.

The results show that the avg. MOTA and FP rate on AAU fulfill the requirements, while the avg. MOTA and FP rate is too low and high on Friis, respectively. The requirements on time consumption is fulfilled for Friis and also largely for AAU, where the time consumption is 16ms and 22ms too high when tracking four and seven targets, respectively. Figure 11.1 shows a visualization of the estimated trajectories and the GT trajectories from the AAU3 sequence. Figure 11.2 shows a visualization of the estimated trajectories and the GT trajectories from the Friis3 sequence.



(a) **(b)**

Figure 11.1: *Visualization of the estimated trajectories and GT trajectories from the AAU3 sequence. Figure (a) shows the GT trajectories. Figure (b) shows the estimated trajectories.*



(a) **(b)**

Figure 11.2: *Visualization of the estimated trajectories and GT trajectories from the Friis3 sequence. Figure (a) shows the GT trajectories. Figure (b) shows the estimated trajectories.*

11.3 Discussion

The results show that the ”Preprocessing” subsystem fulfill the requirement for both AAU and Friis. Furthermore, the results show that the ”Processing” subsystem fulfills the requirements on MOTA and FP rate for AAU and the requirement on time consumption for both AAU and Friis. However, the requirements on MOTA and FP rate are not fulfilled for Friis. By inspecting the estimated trajectories and the GT trajectories from AAU3 and Friis3 in Figure 11.1 and Figure 11.2 it is seen that for AAU, noise is present in the top right corner of the image. For Friis3, significant noise is present in the left edge of the image and in the top right corner. Also, identity switches occur. However, the general patterns of the estimated trajectories resemble the general patterns of the GT trajectories. Thus, it is seen that the ”Preprocessing” and ”Processing” subsystems are able to estimate representative trajectories altough the precision of the Friis trajectories do not fulfill the requirements.

Chapter 12

Acceptance Test

In this chapter, the acceptance test is described. The test is performed in accordance with the Acceptance Test Specification in Chapter 4. The acceptance test consists of two parts; a quantitative test and a qualitative test. Initially, the scene information in AAU and Friis was estimated as described in the following. The "Preprocessing" subsystem was first set to process the AAU2 dataset to estimate the AAU scene information. Likewise, for the Friis data the "Preprocessing" subsystem was set to process the Friis2 dataset to estimate the Friis scene infomation. The estimated scene information was used for both the quantative test and the qualitative test. Having estimated the scene information, the quantitive test was first performed.

12.1 Quantitative

For AAU, the system was set to process the AAU3 dataset using the AAU scene information. For Friis, the system was set to process the Friis3 dataset using the Friis scene information.

In the following, the requirements are listed along with the results of the test for AAU and Friis.

1. Present heat maps of the following statistics (described in Section 2.4):
   a) Person Position with a mean error less than 0.1.
   AAU: 0.0542.
   Friis: 0.0725.
   b) Speed with an SMAPE less than 15%.
   AAU: 5.40%.
   Friis: 5.71%.
   c) Dwell Analysis with a mean error less than 0.1.
   AAU: 0.0033.
   Friis: 0.0050.
2. Present vector plots of the following two statistics (described in Section 2.4):
   a) Mean Directions;
      1. with a mean angle error less than 10°.
      AAU: 5.23°.
      Friis: 4.54°.
      2. with a magnitude SMAPE less than 15%.
      AAU: 4.68%.
      Friis: 5.09%.

b) Top-Four Directions with a SMAPE less than 15%.
AAU: 0.46%.
Friis: 0.67%.

3. Present a Number of Persons plot (described in Section 2.4) with a mean error less than 1.5.
AAU: 0.5714.
Friis: 1.1933.

12.2 Qualitative

In this section, the qualitative acceptance test is described. The qualitative test was first performed on the AAU data and then the Friis data.

12.2.1 AAU

Two qualitative tests on AAU were performed;

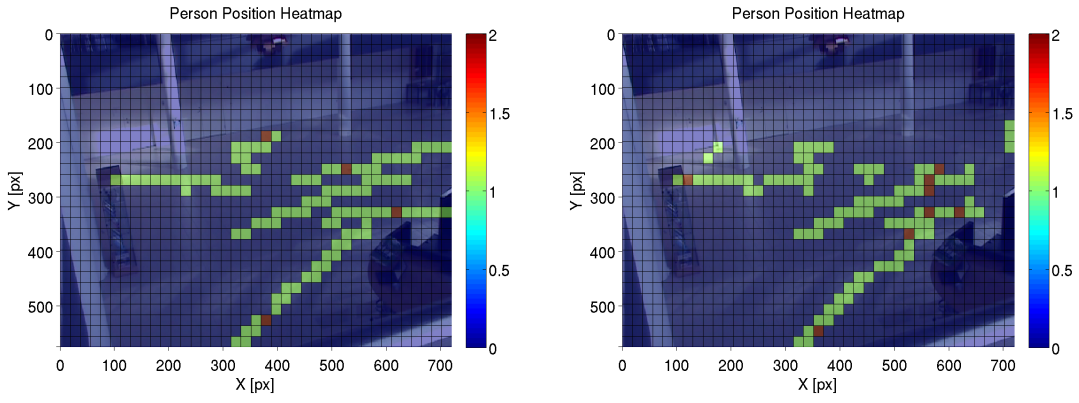
- A test on the AAU3 dataset, which has associated GT.
- A test on the AAU4 and AAU5 dataset, where no GT is available.

The AAU4 sequence has been inspected manually. The plots generated from the AAU4 sequence were then evaluated qualitatively to determine, whether they matched the manual inspection intuitively. The participants in the AAU5 sequence have been instructed to move in a specific pattern, as described in Section 2.6.2. Thus, the plots generated from the AAU5 sequence were evaluated qualitatively to determine, whether these patterns could be recognized.

The system was set to process the AAU4 and AAU5 dataset using the AAU scene information . After this, the system was set to process the AAU3 dataset using the AAU scene information. Finally, the system was set to process the GT for the AAU4 dataset, using the AAU scene information, yielding the GT plots. These sets of plots are evaluated in the following.

Person Position

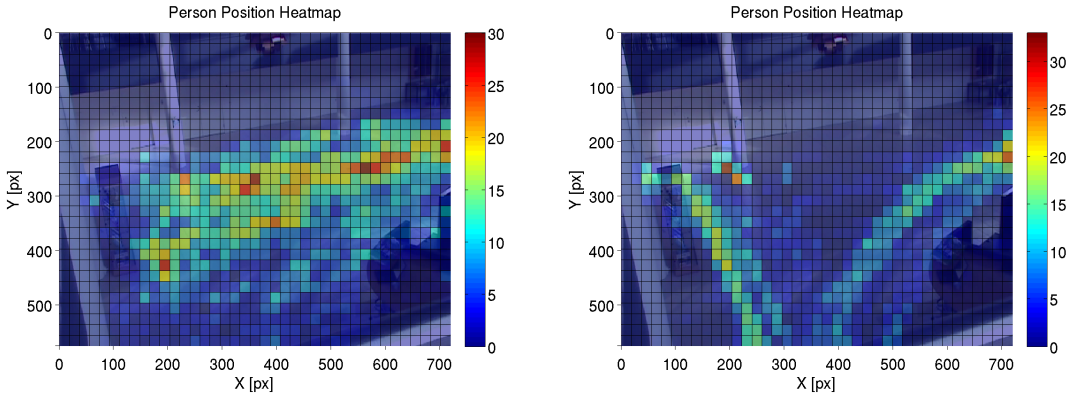
Figure 12.1 shows the Person Position plot generated from AAU3. It is seen in the figure that the estimated plot resembles the GT plot, although with small errors mostly in the top right corner of the image.



(a) **(b)**

Figure 12.1: *Person Position plot generated from AAU3. Figure (a) shows the GT plot. Figure (b) shows the estimated plot.*

Figure 12.2 shows the Person Position plot generated from AAU4 and AAU5. The figure shows that the movement pattern resembles the instructions given to the participants described in Section 2.6.2.

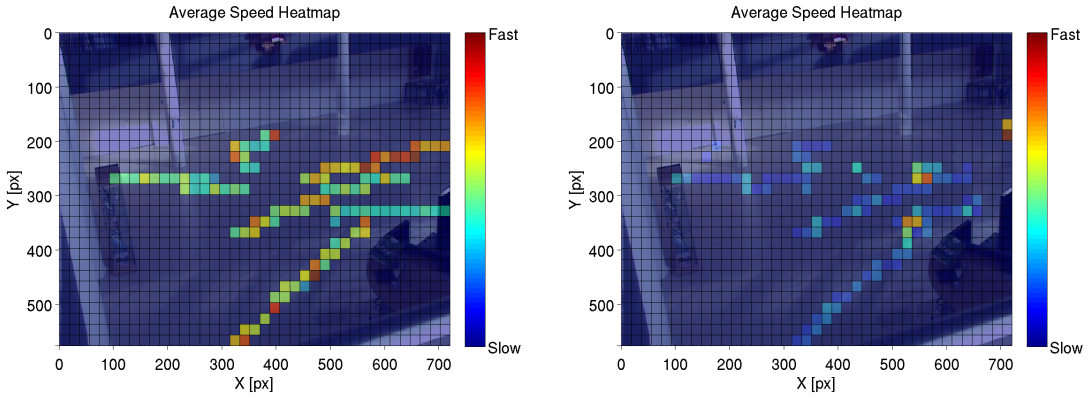


(a) **(b)**

Figure 12.2: *Person Position plots generated from AAU4 and AAU5. Figure (a) shows the AAU4 plot. Figure (b) shows the AAU5 plot.*

Speed

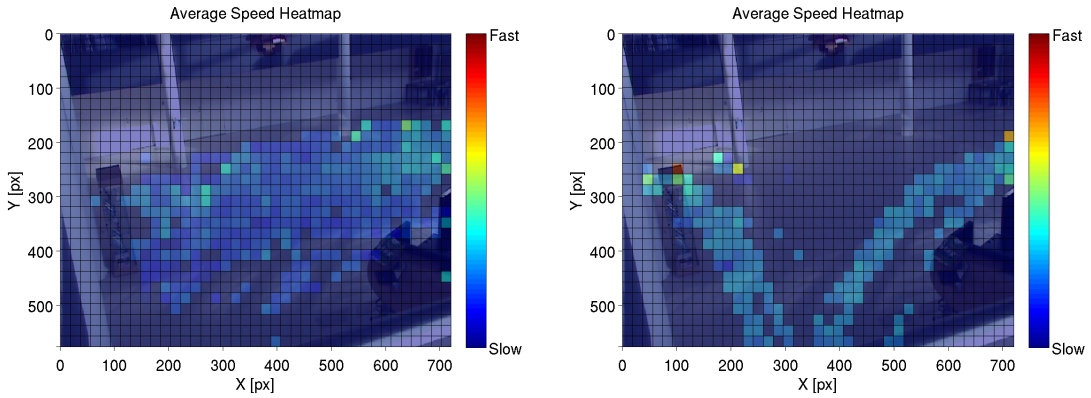
Figure 12.3 shows the Speed plot generated from AAU3. It is seen in the figure that the color of the bins are more red overall in the GT plot which is caused by single bins containing high values due to noise. This result in the color scale being spread over a greater range. Otherwise, the estimated plot resembles the GT plot.



(a) **(b)**

Figure 12.3: *Speed plot generated from AAU3. Figure (a) shows the GT plot. Figure (b) shows the estimated plot.*

Figure 12.4 shows the Speed plot generated from AAU4 and AAU5. The figure shows that a high average speed is estimated in the top right corner of the image. Furthermore, it shows that the participants moved at a roughly constant speed. Both plots resemble the instructions given to the participants.

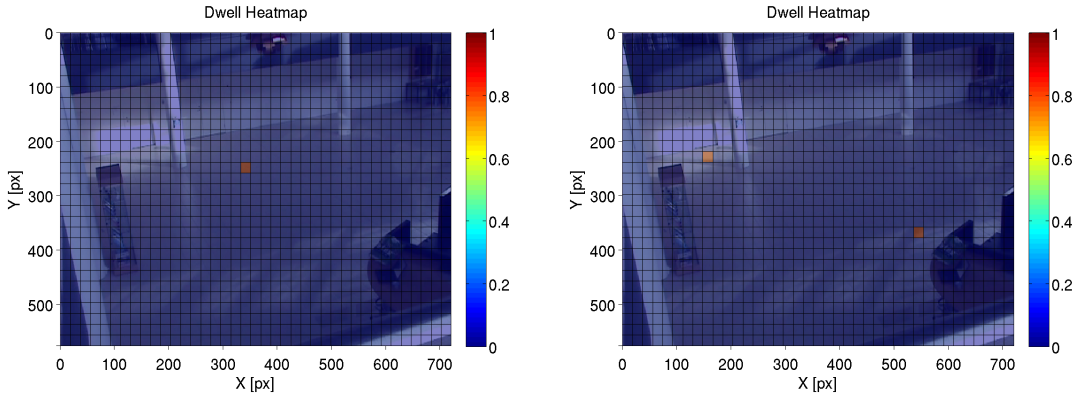


(a) **(b)**

Figure 12.4: *Speed plots from AAU4 and AAU5. Figure (a) shows the AAU4 plot. Figure (b) shows the AAU5 plot.*

Dwell Analysis

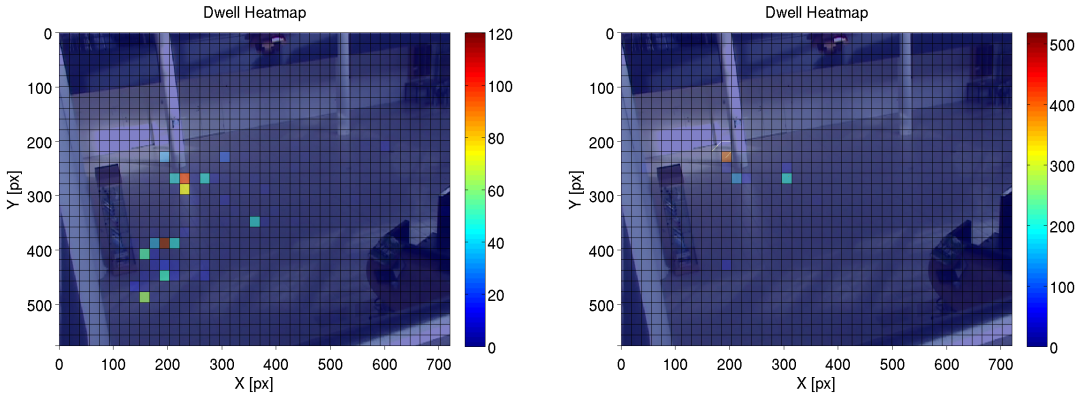
Figure 12.5 shows the Dwell Analysis plot generated from AAU3. It is seen in the figure that the GT plot shows one dwelling sample. The estimated plot does not get this right and has two false positives. The sequence is short, resulting in few dwelling action being counted. Therefore, it is difficult to estimate this statistic correctly.



(a) **(b)**

Figure 12.5: *Dwell Analysis plot generated from AAU3. Figure (a) shows the GT plot. Figure (b) shows the estimated plot.*

Figure 12.6 shows the Dwell Analysis plot generated from AAU4 and AAU5. The figure shows that most people dwell in front of the glass showcase and in front of the pole in the left half of the image. This corresponds to the instructions given to the participants.

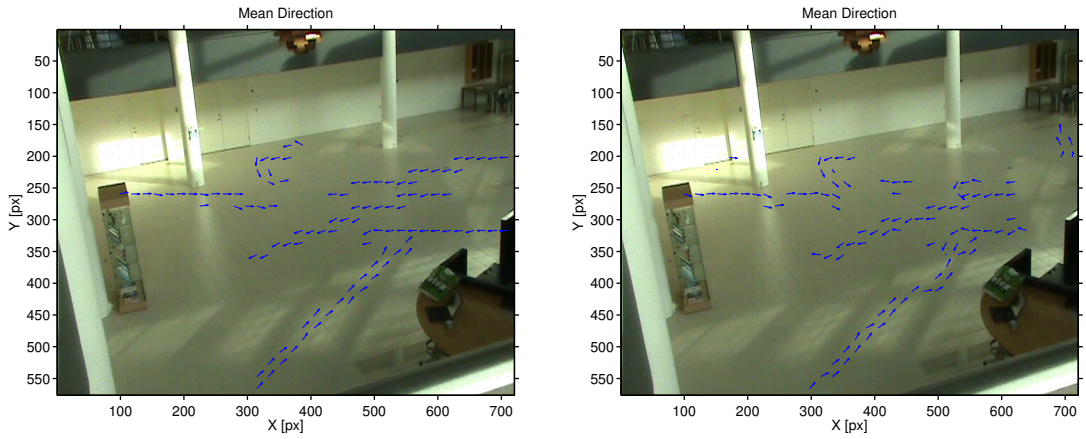


(a) **(b)**

Figure 12.6: *Dwell Analysis plots generated from AAU4 and AAU5. Figure (a) shows the AAU4 plot. Figure (b) shows the AAU5 plot.*

Mean Directions

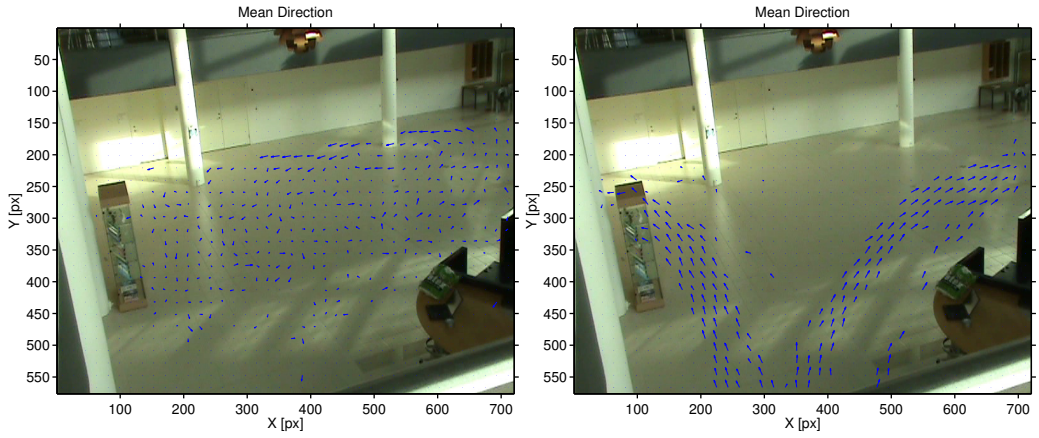
Figure 12.7 shows the Mean Directions plot generated from AAU3. The figure shows that the estimated plot and the GT plot resemble each other. The overall tendency is the same, although with small errors occurring.



(a) **(b)**

Figure 12.7: *Mean Directions plot generated from AAU3. Figure (a) shows the GT plot. Figure (b) shows the estimated plot.*

Figure 12.8 shows the Mean Directions plot generated from AAU4 and AAU5. The figure shows mean directions that corresponds to the instructions given to the participants.

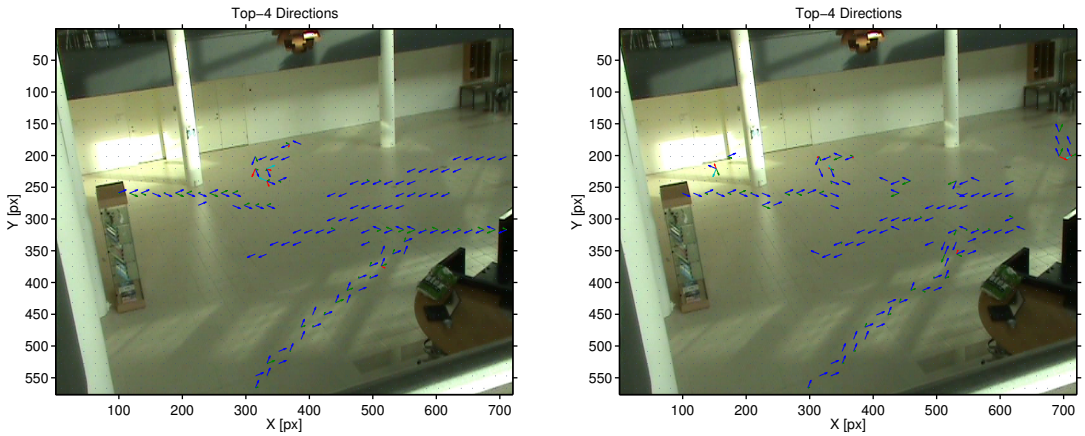


(a) **(b)**

Figure 12.8: *Mean Directions plots generated from AAU4 and AAU5. Figure (a) shows the AAU4 plot. Figure (b) shows the AAU5 plot.*

Top-Four Directions

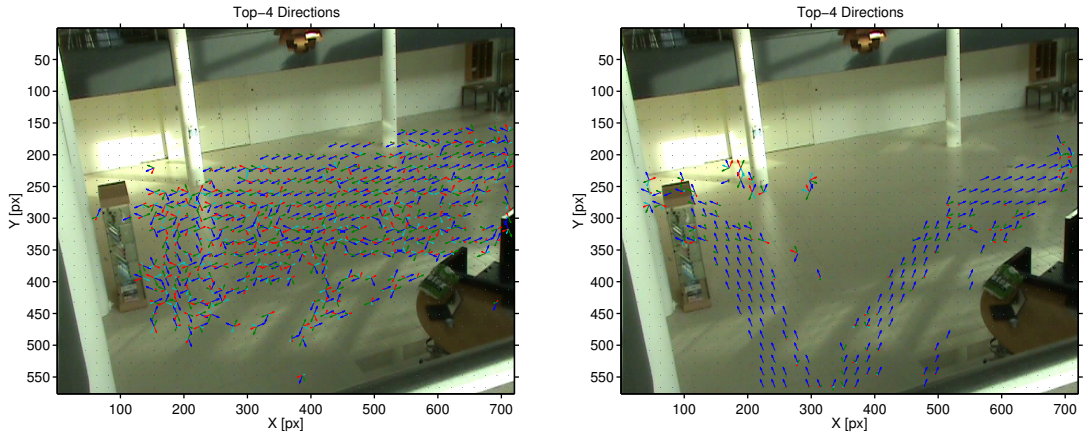
Figure 12.9 shows the Top-Four Directions plot generated from AAU3. As with the Mean Direction plots described before, the figure shows that the estimated plot and the GT plot resemble each other with small errors occurring.



(a) **(b)**

Figure 12.9: *Top-four Directions plot generated from AAU3. Figure (a) shows the GT plot. Figure (b) shows the estimated plot.*

Figure 12.10 shows the Top-Four Directions plot generated from AAU4 and AAU5. The figure shows Top-Four Directions that resemble the instructions given to the participants.



(a) (b)

Figure 12.10: *Top-Four Directions plots generated from AAU4 and AAU5. Figure (a) shows the AAU4 plot. Figure (b) shows the AAU5 plot.*

Number of Persons

Figure 12.11 shows the Number of Persons plot generated from AAU3. The figure shows that the estimated number of people is correct at the beginning. However, as more people enter the scene, a delay occurs in the estimated plot. During most of the period, the estimated number of persons varies around the GT with a difference of one person.

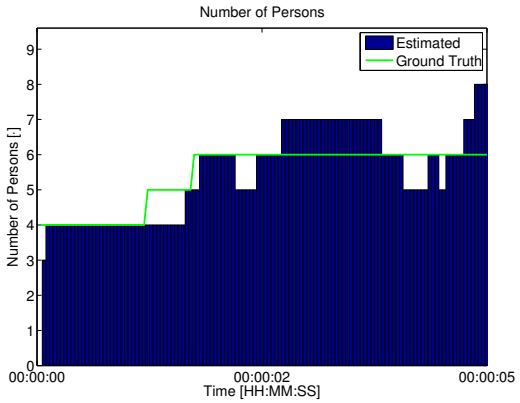
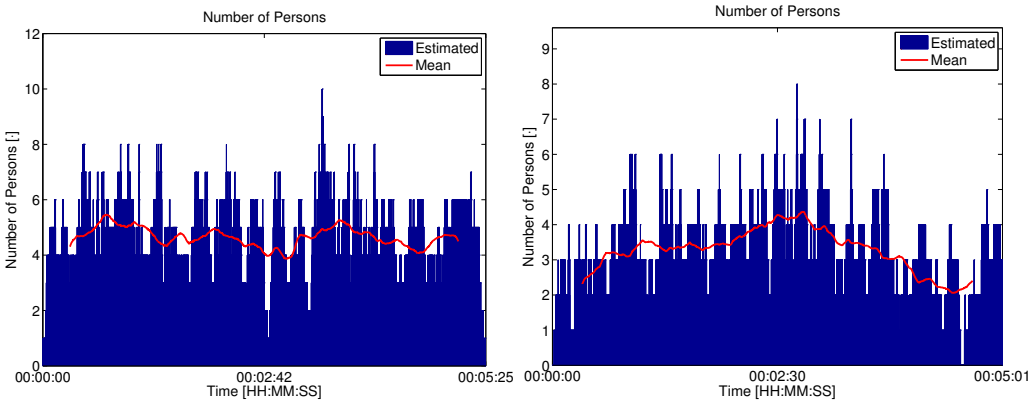


Figure 12.11: *Number of Persons plot generated from AAU3 with the GT shown.*

Figure 12.12 shows the Number of Persons plot generated from AAU4 and AAU5. The figure shows a number of persons that corresponds roughly to the number of persons in the sequences, although the plot in AAU4 (Figure 12.12a) shows a pike with a significant overshoot, i.e. 10 persons estimated but only seven participants.



(a) **(b)**

Figure 12.12: *Number of Persons plots generated from AAU4 and AAU5. Figure (a) shows the AAU4 plot. Figure (b) shows the AAU5 plot.*

12.2.2 Friis

Two qualitative tests on Friis were performed;

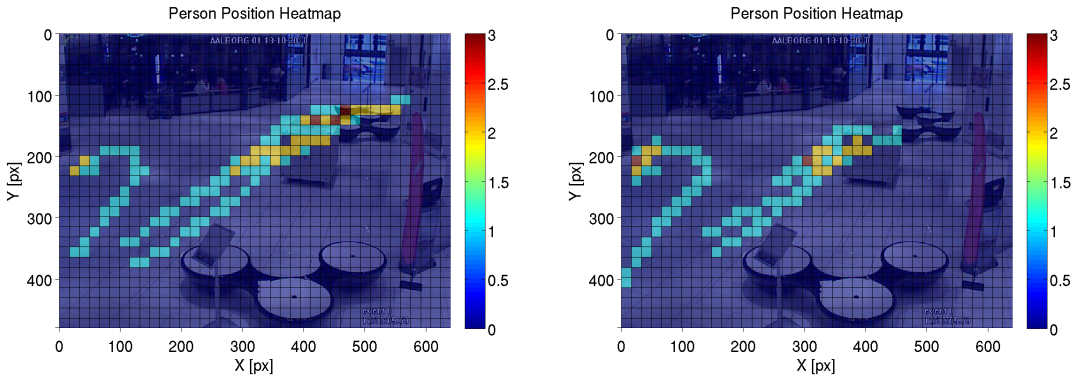
- On the Friis3 dataset, which has associated GT.
- On the Friis4 dataset, where no GT is available.

The Friis4 sequence has been inspected manually. The plots generated from the Friis4 sequence were then evaluated qualitatively to determine, whether they matched the manual inspection of the video sequences.

The system was set to process the Friis4 dataset using the Friis scene information. After this, the system was set to process the Friis3 dataset using the Friis scene information. Finally, the system was set to process the GT for the Friis3 dataset using the Friis scene information, yielding the GT plots. These sets of plots are evaluated in the following.

Person Position

Figure 12.13 shows the Person Position plot generated from Friis3. The figure shows that the estimated plot and the GT plot resemble each other, although the system has not captured the movement at the top right corner of the image.



(a) **(b)**

Figure 12.13: *Person Position plots generated from Friis3. Figure (a) shows the GT plot. Figure (b) shows the estimated plot.*

Figure 12.14 shows the Person Position plot generated from Friis4. The figure shows high values close to the left border of the image. This corresponds to the manual inspections of the sequence, where it is seen that most people enters the image from the left side and exits through the left border shortly after. Furthermore, the figure shows high values close to the box in the middle of the image, which corresponds to the inspections that showed many people walking close the box to look closer and to read the posters in the right side of the image. However, the inspections also showed a relatively low number of persons walking through the top right corner of the image. This has not been captured by the system.

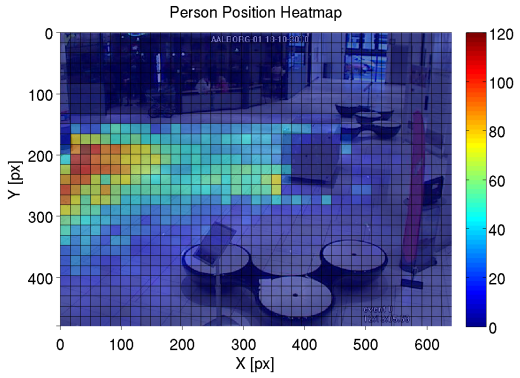
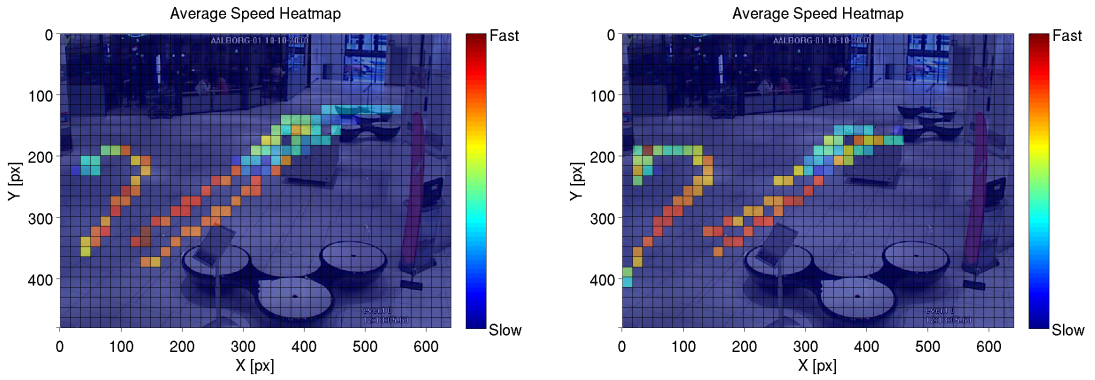


Figure 12.14: *Person Position plot generated from Friis4.*

Speed

Figure 12.15 shows the Speed plot generated from Friis3. The figure shows at the estimated plot and the GT plot resemble each other. However, the system has not estimated the correct speeds in the top right corner of the image.



(a) **(b)**

Figure 12.15: *Speed plots generated from Friis3. Figure (a) shows the GT plot. Figure (b) shows the estimated plot.*

Figure 12.16 shows the Speed plot generated from Friis4. The figure shows that people have moved at a roughly constant speed, although a lower speed is estimated around the box in the middle of the image. This corresponds with the manual inspections of the sequence. However, a higher speed is estimated at the top left corner of the image, which did not show in the inspections. Also, no average speeds above zero have been estimated for the bins in the top right corner of the image, which does not correspond with the inspections.

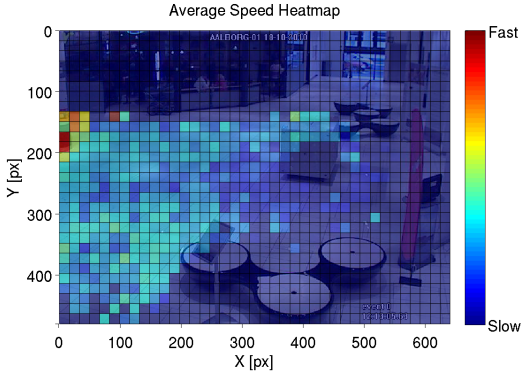
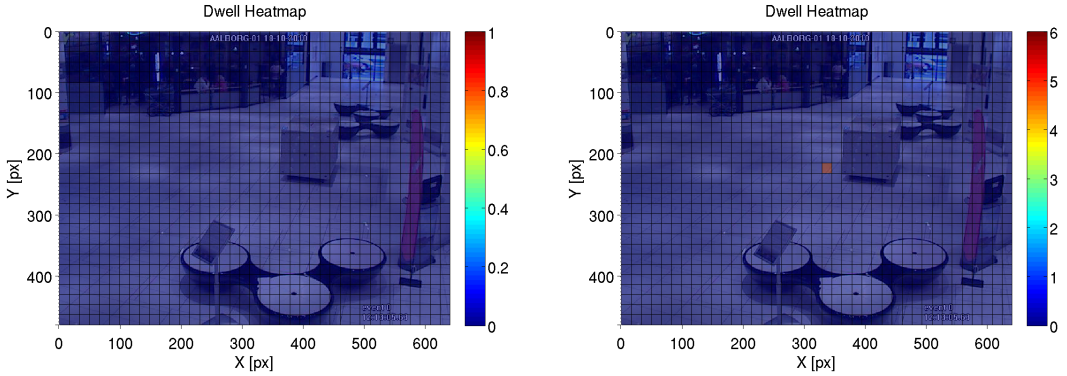


Figure 12.16: *Speed plot generated from Friis4.*

Dwell Analysis

Figure 12.17 shows the Dwell Analysis plot generated from Friis3. The GT in Figure 12.17a shows that no people dwells during the sequence. An inspection of the sequence shows that people dwell in front of the box at the end of the sequence. However, the dwelling happens in the last frames of the sequence and is removed in the preprocessing step in the ”Visualization” module by the smoothing kernel, see Section 10.2. The estimated heat map in Figure 12.17b shows that the system has estimated a number of persons dwelling in front of the box, thus corresponding to what happens in the sequence. This is however caused by a small noise in the estimations.



(a) **(b)**

Figure 12.17: *Dwell Analysis plots generated from Friis3. Figure (a) shows the GT plot. Figure (b) shows the estimated plot.*

Figure 12.18 shows the Dwell Analysis plot generated from Friis4. The figure shows that many people dwell near the box in the middle of the image. This corresponds with the inspections of the sequence, where many people walk up to the box to look closer and to read the posters in the right side of the image.

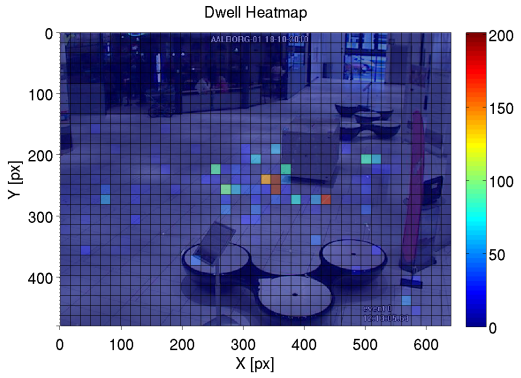
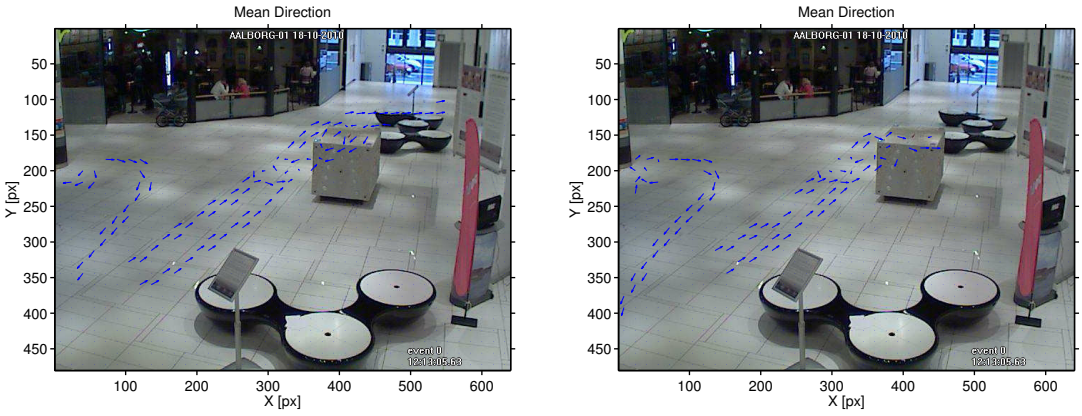


Figure 12.18: *Dwell Analysis plot generated from Friis4.*

Mean Directions

Figure 12.19 shows the Mean Directions plot generated from Friis3. The figure shows that the GT and the estimated heat map resemble each other, although the estimated directions in the top right corner of the GT plot are not present in the estimated plot.



(a) **(b)**

Figure 12.19: *Mean Directions plots generated from Friis3. Figure (a) shows the GT plot. Figure (b) shows the estimated plot.*

Figure 12.20 shows the Mean Directions plot generated from Friis4. The figure shows that people generally do not have a preferred direction, i.e. it cannot be determined, in which direction most people move. This corresponds with the inspections of the sequence.

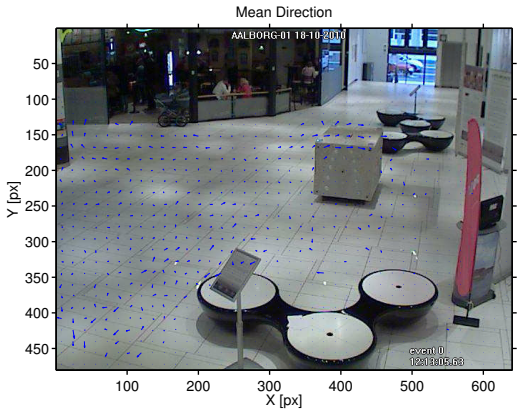
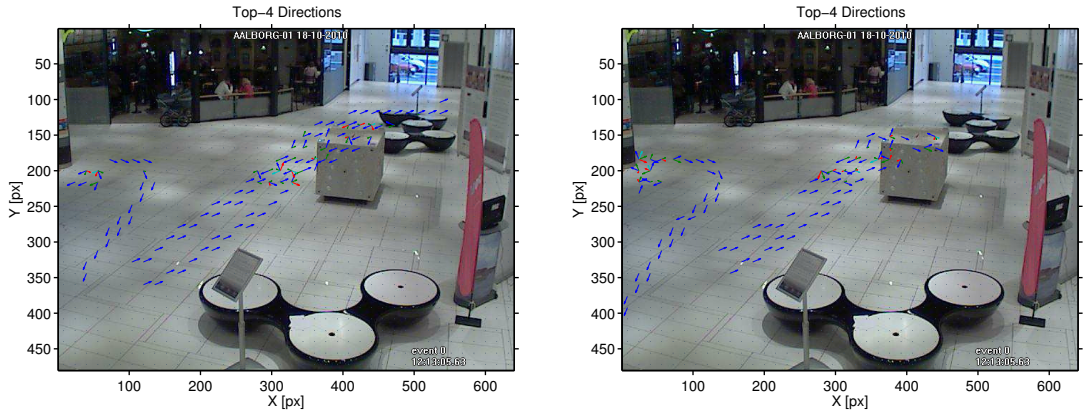


Figure 12.20: *Mean Directions plot generated from Friis4.*

Top-four Directions

Figure 12.21 shows the Top-Four Directions plot generated from Friis3. The figure shows that the GT plot and the estimated plot resemble each other, although with errors in the estimated plot in the top right corner of the image.



(a) (b)

Figure 12.21: *Top-four Directions plots generated from Friis3. Figure (a) shows the GT plot. Figure (b) shows the estimated plot.*

Figure 12.22 shows the Top-Four Directions plot generated from Friis4. The figure shows the four arrows representing the most frequent directions are generally equally long. As with the Mean Directions, this indicates that people generally do not have a preferred direction, which corresponds with the inspections of the sequence.

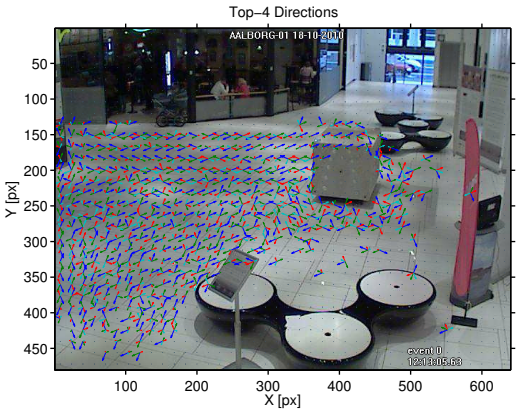


Figure 12.22: *Top-Four Directions plot generated from Friis4.*

Number of Persons

Figure 12.23 shows the Number of Persons plot generated from Friis3. The figure shows that the estimated plot resembles the GT plot. However, timing errors occur in the estimated plot, where the estimated plot is behind the GT plot in the beginning, and ahead in the second half of the plot. The difference between the GT and the estimated plot is generally one person, with a larger error in the beginning of the plot.

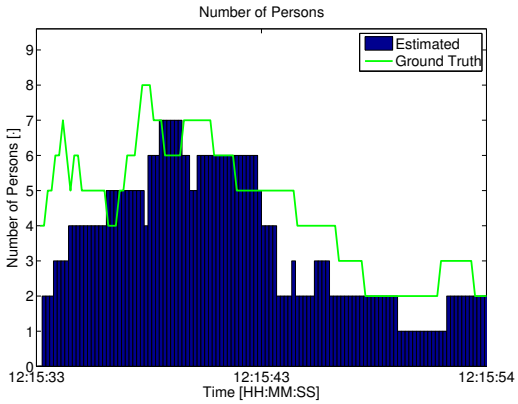


Figure 12.23: *Number of Persons plot generated from Friis3, with the GT shown.*

Figure 12.24 shows the Number of Persons plot generated from Friis4. The figure shows that the number of persons fluctuates significantly and rapidly over time. The estimated number changes regularly from values between zero and two to values between eight and 12 rapidly. This corresponds with the inspections of the sequence, which showed several groups of people entering and leaving the scene than single persons entering and leaving.

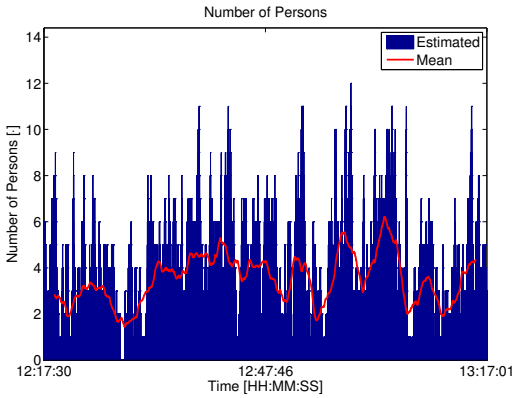


Figure 12.24: *Number of Persons plot generated from Friis4.*

12.3 Discussion

An overview of the results from the quantitative test is given in Table 12.1.

It is seen that all requirements are fulfilled for the quantitative test. For the quantitative test, small datasets were used, i.e. the Friis3 dataset consisting of 121 frames and AAU3 consisting of 124 frames. For a more representative quantitative test, significantly larger datasets are necessary. For larger datasets, more targets go through each bin and give more samples for estimating a more

Statistic	Friis error	AAU error	Requirement
Position.	0.0725	0.0542	0.1
Speed.	5.71 %	5.40%	15%
Dwell.	0.0050	0.0033	0.1
Mean Directions; angle.	4.54°	5.23°	10°
Mean Directions; magnitude.	5.09%	4.68%	15%
Top-Four Directions.	0.67%	0.46%	15%
Number of Persons.	1.1933	0.5714	1.5

Table 12.1: *Overview of results from the quantitative test. Results that do not fulfill the requirements are written in bold.*

representative the value. However, because of the time needed to annotate larger sets of GT for such sequences, it has not been possible to create larger datasets.

In the qualitative test, it was seen that the estimated plots generally resembled the GT plots. Furthermore, for the Friis data the estimated plots corresponded to the manual inspections of the sequence. For the AAU plots the estimated plots corresponded to the instructions given to the participants. However, for both the AAU and Friis sequences, errors that were consistent through all the estimated plots were noticed.

For Friis, most errors were present in the top right corner of the image, where no movement was detected. Inspection of the sequences show that persons walking in the top right corner of the image are so small that the "Human Detector" module cannot detect them. However, persons walking in the top right corner of the image are far away from the surveillance camera in question. This visible area is overlapped by another surveillance camera, which has a better view of this area.

For AAU, most errors were present near the left pole in the image, where noise corrupted the estimations. Here, the "Human Detector" has problems making correct detections, i.e. the area has structures that resemble humans according to the "Human Detector".

Thus it can be concluded that the system is able to generate plots, where most of them fulfill the requirements. A qualitative test showed errors that were consistent in the Friis plots and errors that were consistent in the AAU plots. For Friis, the system cannot detect movements that are far away. For AAU, a specific area in the scene has structures that resemble humans. However, all generated plots resemble either the GT, the inspections given to the participants or the manual inspections of the data.

Chapter 13

Conclusion

The goal of this project was to develop a system that could generate statistics on the movement of people inside Friis for commercial purposes. First, the problem of generating statistics on the movement of people was investigated thoroughly in the Analysis. For this, the use of surveillance was analyzed and a list of the most relevant movement statistics and how to represent them was set up. Through a comparison of existing products and technologies, it was chosen to use the existing surveillance cameras in Friis together with computer vision to solve the task. By analyzing different abstraction levels, it was chosen to generate statistics on movement of people by detecting and tracking people over time.

Related work in the area of tracking people in video sequences was investigated. It was chosen to use a method proposed Breitenstein et al.[21] as this method was found to be the most appropriate for this work. It was decided to modify this method by making the system estimate and use scene information to improve the performance and also by generating and representing the movement statistics visually. The data used for developing and evaluating the system was described. Besides from using surveillance footage from Friis, video sequences that were recorded specifically for this work at AAU were used. A few limitations were set up, before the specific problem statement was defined. This led to a Requirement Specification and an Acceptance Test Specification.

After this, the Program Design was defined. Here, the system was split into three subsystems, for which the external interfaces were defined. The three subsystems are ”Preprocessing”, which estimates scene information from a long sequence, ”Processing”, which estimates trajectories of persons, while using the scene information to improve its performance and ”Visualization”, which extracts the movement statistics from the estimated trajectories and makes a visualization of these. Each subsystem was then split into modules and the requirements for each module were defined. This was followed by a test specification for the subsystems.

The modules were designed and evaluated separately and described in the chapters following the Program Design. After this, the subsystems and the entire system were evaluated in the Subsystem Test and in the Acceptance Test, respectively. The results of the Subsystem Test showed that the system was able to estimate the scene information accurately and estimate trajectories satisfactorily, although not for the Friis data, where the precision was significantly lower than for the AAU data. The Subsystem test also showed a time consumption that is lower than the time consumption reported by Breitenstein et al.[21] For the Acceptance Test, the statistics generated from the AAU data was significantly more accurate than the statistics generated from the Friis data. However, a qualitative evaluation of the system showed that the estimated statistics highly resembled the ground truth statistics both for the AAU and Friis datasets. Furthermore, for Friis it was seen that the system's inability to detect distant persons in a specific area of image contributed to the errors of the estimated statistics. The area in Friis, where people become too distant for the system to handle, is however also covered by other surveillance cameras that are closer than the camera used to record the data for this work.

Thus, a system that can generate statistics on the movement of people inside Friis has been developed. Although the measured precision of the estimated statistics is lower for the Friis data than for the AAU data, the estimated statistics still represent the actual people movement to a satisfactory degree. The system works under the following assumptions:

- The surface, on which people are being tracked must be planar.
- It must be possible to model the size of people as a function of image coordinates as a plane.
- A known focal length or a rough estimate of the focal length must be available.
- The frame rate must be constant.

This work makes the following contribution to the problem of tracking multiple persons in complex scenes: enhancing the output from human detectors by exploiting scene information automatically generated from reliable human detections. Furthermore, this work extract relevant statitics and represent them visually.

Suggestions on future work for improving the system are given in the following section.

13.1 Future Work

This section describes possible improvements and extensions of the system.

It was found in Chapter 6 that the time consumption of the Human Detector module was reduced by using the estimated scene information. The time consumption by this module might be reduced further by taking advantage of GPU processing. For this, GPU implementations of HOG-detectors already exist.[40][45]

It was seen in the Subsystem Test and in the Acceptance Test that the system could not detect persons in Friis that are too distant. An improvement of the system will therefore be to make the Human Detector able to detect persons of sizes that are even smaller than the current minimum size of persons that can be detected.

The "Multi-Human Tracker" can be improved by making a more efficient method for terminating trackers. Inspections of the "Multi-Human Tracker" has shown that trackers are regularly terminated prematurely. However, making the module more hesitant towards termination of trackers introduces new problems when targets leave the scene as the trackers remain active for too long, thus increasing the FP rate of the module. The module needs to be improved on this area.

An extension of the system will be to make it run on multiple cameras, and possibly to make it able to track people across cameras. The latter will potentially increase the robustness of the "Multi-Human Tracker", as the tracker might have been trained for a person that is leaving one camera and entering the other camera. Instead of terminating the tracker for that person and initializing a new, the tracker can then remain running as the person enters the other camera and thus the tracker will have more knowledge on the appearance of the person than a newly initialized tracker would. Another extension of the system would be to generate new kinds of statistics from the estimated trajectories, potentially expanding the application domain of the system. Also, an extension of the system would be to implement the intermediate outputs and automatic detection of behavior change, described in the "Movement Statistics", Section 2.4.

Thus, suggestions on improving the system in terms of time consumption and accuracy have been given. Also, a suggestion on how to extend the system has been given. The most important areas of future work are improving the way the "Multi-Human Tracker" module handles the termination of trackers.

Appendix A

Test and Evaluation Procedure

In this appendix, the procedures for testing and evaluating the "Human Detector" and "Multi-human Tracker" modules are described. The two test procedures have similarities, and therefore they are both described in this appendix.

A.1 Human Detector

In this section, the procedure for testing and calculating a measure of the performance of the output from the "Human Detector" module is described. The "Human Detector" module is tested quantitatively using GT data. The GT data contains information about size and position of all humans in each frame and is generated manually. In each frame, all present humans according to the GT data, are matched against the detected humans from this module to examine how many of the detections that are correct or wrong.

Since the GT data is extracted by manually defining the position of humans in videos, their positions can be imprecise. Therefore, a margin of error when matching a detection to a GT detection is allowed. The matching is done by evaluating all detections in a frame with all the GT detections in the same frame with a scoring function:

$$S(d,g) = \Delta_p(d,g) + \Delta_s(d,g) \qquad \text{for } d \in D_t, g \in G_t \qquad [\cdot] \quad \text{(A.1)}$$

where:

D_t:	Set of detection for the frame at time t.	$[\cdot]$
G_t:	Set of GT detections for the frame at itme t.	$[\cdot]$
$\Delta_p(d,g)$:	Difference of position between detection d and GT g.	$[\cdot]$
$\Delta_s(d,g)$:	Difference of size between detection d and GT g.	$[\cdot]$

The $\Delta_p(d,g)$ and $\Delta_s(d,g)$ functions are normalized with respect to the size of the GT detection so a fixed threshold can be used to confirm a match with detections at all sizes. The functions are given by:

$$\Delta_p(d,g) = \sqrt{\left(\frac{x_d - x_g}{w_g}\right)^2 + \left(\frac{y_d - y_g}{h_g}\right)^2} \qquad [\cdot]$$

$$\Delta_s(d,g) = \left|\frac{w_g \cdot h_g - w_d \cdot h_d}{w_g \cdot h_g}\right| \qquad [\cdot]$$

where:

x_d, y_d: Coordinates of the detection center. [·]
x_g, y_g: Coordinates of the GT center. [·]
w_d, h_d: Width and height of the detection. [·]
w_g, h_g: Width and height of the GT detection. [·]

(A.2)

A match is made between a detection and a GT detection if the pair has the lowest score of all the detection and GT pairs at a given frame. Furthermore, $\Delta_p(d, g)$ must be below 0.4 while $\Delta_s(d, g)$ has to be below 0.3. These thresholds have been found through experiments to yield good matching decisions.

If a detection has a matching GT, the detection is counted as a true positive but if no GT matches the detection, it is counted as a false positive. All GT with no matching detections are counted as false negatives. The types of errors are illustrated in Figure A.1.

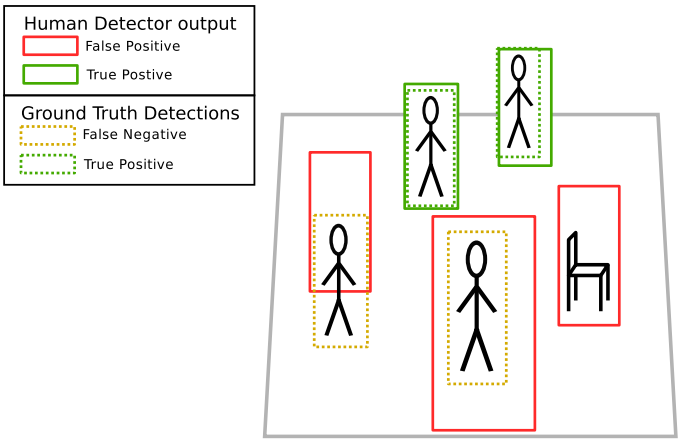


Figure A.1: *A scene with four humans and four GT detections (dashed rectangles). The "Human Detector" has detected five humans (solid rectangles). However, three of them are wrong (red rectangles) while two are correct and therefore matched to GT detections (green). The GT detections that have no matched detections are False Negatives (yellow rectangles).*

The number of False Positives, True Positives and False Negatives at frame t are denoted as: (FP_t) (TP_t) (FN_t). The accuracy of the "Human Detector" is evaluated on these three parameters by:

$$\mathrm{FP} = \frac{\sum_t \mathrm{FP}_t}{n_d} \cdot 100 \quad [\%]$$

$$\mathrm{FN} = \frac{\sum_t \mathrm{FN}_t}{n_g} \cdot 100 \quad [\%]$$

$$\mathrm{TP} = \frac{\sum_t \mathrm{TP}_t}{n_g} \cdot 100 \quad [\%]$$

where:

FP_t: Number of False Positives at frame t. [·]
FT_t: Number of False Negatives at frame t. [·]
TP_t: Number of True Positves at frame t. [·]
n_d: Total number of detections. [·]
n_g: Total number of GT detections. [·]

In each frame, the sum of all the euclidean distances from the matched detection to their GT is calculated and denoted as e_t. The precision of the "Human Detector" is then calculated using this distance error averaged over all matches.

$$P = \frac{\sum_t e_t}{n_m} \qquad [\text{px}]$$

where:

e_t: Sum of errors in pixels at time t. [·]
n_m: The number of matches. [·]

A.2 Multi-Human Tracker

In this section, the procedure for testing and measuring the performance of the "Multi-Human Tracker" module is described. The module is tested like the "Human Detector" module with GT data. The GT data used for testing the "Human Detector" is upgraded with information about which detection that belongs to which target and is then used for testing the "Multi-Human Tracker".

The procedure for checking if the "Multi-Human Tracker" estimates correct trajectories is similar to evaluating the detections from the "Human Detector". In each frame, all trajectory points that consist of a position, size and target ID, are compared to all GT trajectory points in the same frame. This comparison is done by evaluating the scoring function in Equation A.1 and matching the pairs with the same procedure as in the previous section. This function can be used since the trajectories are represented in the same way (i.e with size and position). All tracker and GT points that are matched are counted as true positives. If a GT point is not matched, it is counted as false negative, and if a tracker point is not matched it is counted as a false positive.

For now, the output of the "Multi-Human Tracker" has only been evaluated for the position and size. However, the trackers output also includes the ID of the targets. This ID must be the same for the same target throughout the sequence that it appears in. This is evaluated by monitoring the trajectory point IDs between frames. After all matches have been performed between the GT and tracker points, their corresponding IDs are recorded. In the next frame, the pair with the same IDs should be made. If that is not the case, a mismatch is noted and the new relation is recorded.

The three types of errors are illustrated in Figure A.2. Note that the trajectory points also include the size of the target. However, for simplicity only the position and tracker ID are shown in the figure.

For evaluating the "Multi-Human Tracker", the CLEAR MOT metrics [39] are used. They is used among others by Breitenstein et al. The CLEAR MOT metrics are designed to be an evaluation measure for generic object tracking and measure on two parameters; the multi object tracking precision (MOTP) and the multi object tracking accuracy (MOTA). The MOTP expresses how the position of the tracker corresponds to the actual position of the target, while the MOTA express how many false positive, false negatives and the mismatches made by the trackers.

The number of False Positives, False Negatives and mismatches are counted at each frame and are denoted as: FP_t, TP_t and FN_t, respectively. MOTA is calculated by:

$$\text{MOTA} = \frac{\sum_t (\text{FP}_t + \text{FN}_t + \text{MM}_t)}{\sum_t n_t} \qquad [\cdot]$$

where:

FP_t: Number of False Positives at frame t. [·]
FN_t: Number of False Negatives at frame t. [·]
MM_t: Number of mismatches at frame t. [·]
n_t: The number of GT detections at frame t. [·]

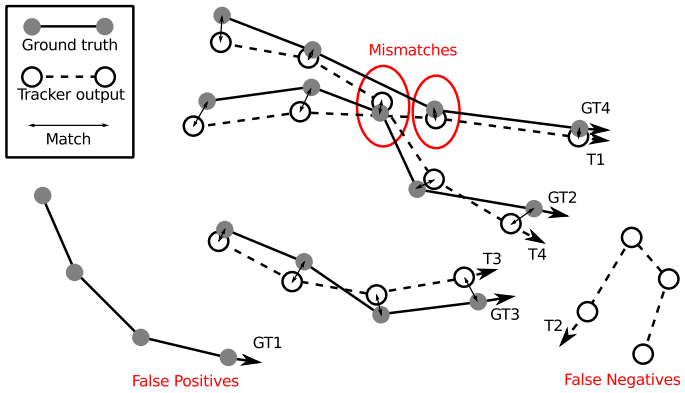


Figure A.2: *An illustration of the three types of errors that the tracker can make. Actual positions of humans are drawn with gray dots and their actual trajectories are drawn with dashed lines, while the points estimated by the trackers are drawn with white dots. For simplicity, the sizes of the targets are not illustrated in this figure.*

MOTP is calculated as described in the following. The sum of errors in the position of all matched pairs at time t is denoted by e_t. The error is calculated as euclidean distances from the trajectory positions estimated by the tracker to the GT. MOTP is then calculated by:

$$MOTP = \frac{\sum_t e_t}{\sum_t m_t} \qquad [\cdot]$$

where:

e_t: The sum of distances from all matched detections to their GT. $[\cdot]$
m_t: The number of matched detections at time t. $[\cdot]$

Appendix B

AdaBoost

AdaBoost is a meta-algorithm, which is able to combine a number of weak classifiers into a strong classifier.[34, p. 497] A weak classifier uses simple features and is not often not able to discriminate the classes alone. AdaBoost is trained supervised and iteratively by selecting the best of the weak classfiers to classify the weighted training samples. AdaBoost is trained supervised with the traning samples given by:

$$S = ((\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)) \quad [\cdot]$$

where:

$\mathbf{x}_i$: The feature vectors of the training data. $[\cdot]$

y_i: The corresponding class of the feature vector $\in \{-1, 1\}$. $[\cdot]$

n: The number of training samples. $[\cdot]$

Each training sample i at iteration t is given a weight $\lambda_t(i)$, which can be seen as a cost of a wrong classification. Before the training starts, the weight is initialized uniformly to:

$$\lambda_1(i) = \frac{1}{n},\ i = 1, \ldots, n \quad [\cdot]$$

After this, all the classifiers $h_j(\mathbf{x})$, $j = 1, \ldots, m$ are trained and the classifier with the lowest training error is selected as the classifier of iteration t: $h_t(\mathbf{x})$. The training error is calculated from the misclassified samples and their weights by:

$$\epsilon_t = \sum_{i=1}^{m} \lambda_t(i) |h_t(\mathbf{x}_i) - y_i| \quad [\cdot]$$

The selected classifier $h_t(\mathbf{x})$ gets a weighted vote α_t used for the final AdaBoost classifier. It is calculated from the training error:

$$\alpha_t = \frac{1}{2} \log \left[\frac{1 - \epsilon_t}{\epsilon_t} \right] \quad [\cdot]$$

The weights of the training samples are updated in a way so that the samples that were misclassified from the selected classifier get a higher weight. This motivates the next iteration to find the classifier that classifies the difficult samples correctly.

$$\lambda_{t+1}(i) = \frac{\lambda_t(i) \exp(-\alpha_t y_t h_t(x_i))}{Z_t},\ i = 1, \ldots, n \quad [\cdot]$$

where:

Z_t: Normalization factor to make the sum of all weight equal to 1. [·]

This is repeated until all weak classifiers are used or until the lowest error of the remaining classifiers is greater than 0.5. Then, a random guess between the two classes is better than the classification. The AdaBoost training algorithm is summarized in Algorithm 2.

Algorithm 2 AdaBoost training algorithm

m {Number of weak classifiers}
$\lambda_1(i) \leftarrow \frac{1}{n}$, $i = 1, \ldots, n$
for $t = 1, \ldots, T$ **do**
  $h_t \leftarrow \arg\min_{h_j} \sum_{i=1}^{m} \lambda_t(i)|h_t(\mathbf{x}_i) - y_i|$
  if $\epsilon_t < 0.5$ **then**
    Stop
  end if
  $\alpha_t \leftarrow \frac{1}{2} \log\left(\frac{(1-\epsilon_t)}{\epsilon_t}\right)$
  $\lambda_{t+1}(i) \leftarrow \frac{\lambda_t(i) \exp(-\alpha_t y_i h_t)}{Z_t}$
end for
return $\mathbf{h}, \alpha$

When using the trained AdaBoost for classification, a sum of the output from the weak classifiers, multiplied with their weights, is calculated. If the result is greater than 0, the samples are classified as positive and if it is less, the samples are classified as negative.

$$H(\mathbf{x}) = \sum_{t=0}^{T} \alpha_t h_t(x) \qquad [\cdot]$$

Appendix C

Target-specific classifier

In this appendix the classifier used for representing the appearance of a target used in the "Data Association" module and for the observation model of the particle filter is described. The classifier used is an onlined learned classifier saved in each tracker and learned on the target being tracked. The features used for the classifier represent the appearance of the target.

C.1 Online Learned Adaboost for Feature selection

Since the targets can change in appearance, e.g. due to change in pose or occlusion, the classifier needs to be able to adapt. Furthermore the appearance of the target is not known in advance, so the classifier needs to learn it online. For this, Breitenstein et al. have proposed to use an online learned AdaBoost algorithm for feature selection proposed by Grabner et al. [42] The algorithm constructs a strong classifier by combining a selection of weak classifiers weighted so that the best classifiers count the most.

In Appendix B, the original AdaBoost algorithm is described, which was first proposed by Freund and Schapire.[46] This algorithm is made for offline training, since it assumes that all training samples are available at an offline training phase. A sample for training is given by $(\mathbf{x}, y)$, where $\mathbf{x}$ is the feature vector and y is the class that it belongs to.

The training process of the original AdaBoost is outlined in Figure C.1 and described in the following. The weak classifiers are trained with all the samples, and their performance is evaluated with the same samples. The best classifier is selected and removed from the weak classifier pool. Each sample gets a sample weight assigned, which expresses how difficult a sample is to classify. The sample weight is used to encourage the classifiers to classify difficult samples correctly. The samples are again classified with the remaining classifiers and their performance is yet again evaluated. The best classifier is selected and the sample weights is updated. This is done until all classifiers are used or untill the error of the best classifier is above 0.5. If the error is above 0.5, a random guess between the two classes is better than the classification.

This training scheme is not designed for a classifier, which has to be learned online like for this application. Here, one sample is extracted and used for updating the classifier. After this, the classifier will never see it again. The AdaBoost proposed by Grabner et al. is able to train the classifiers with samples only shown to it once. This is done by updating the weak classifiers one at a time with one sample. Between each update, the sample is classified with the one sample and depending on the result, the sample weight is updated. The algorithm records the performance of each classifier for all samples used for training and combining the weak classifiers into a strong classifier. An outline of the online learned AdaBoost is illustrated in Figure C.2.

Another thing that makes the AdaBoost proposed by Grabner et al. [42] differ from the original, is that it only uses the best subset of all the weak classifiers. This is done by introducing a "selector". Each selector, $h_n^{sel}(\mathbf{x})$, consists of M randomly picked weak classifiers, $(h_{n,1}^{weak}(\mathbf{x}), \ldots, h_{n,M}^{weak}(\mathbf{x}))$.

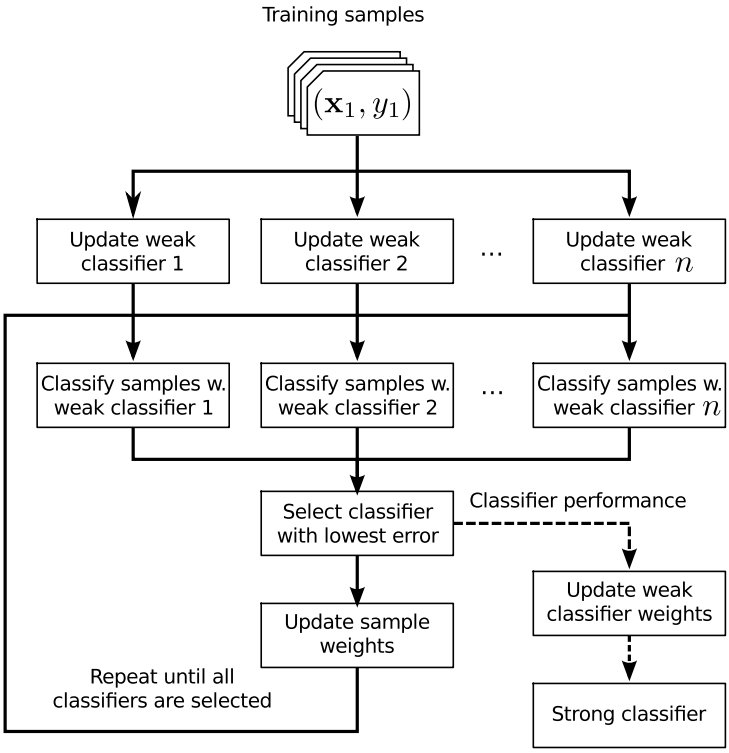


Figure C.1: *Sketch of the training scheme for the offline AdaBoost, where all training samples are used over a number of iterations.*

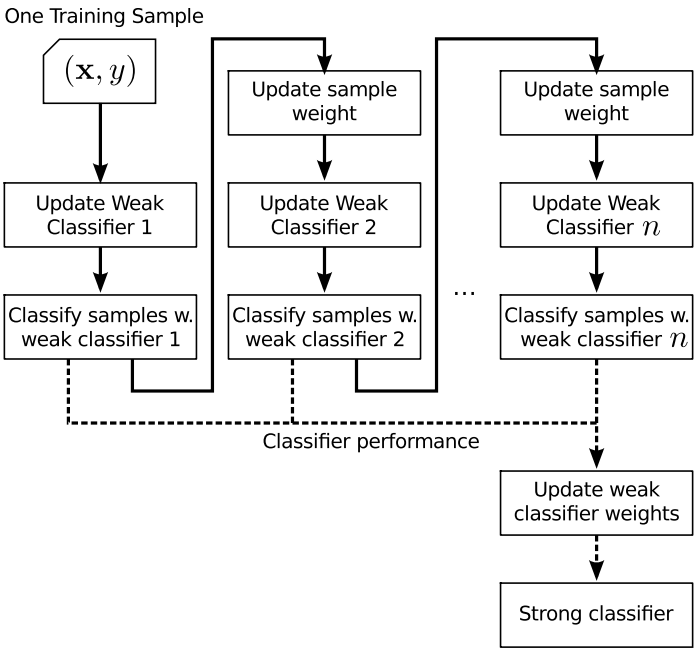


Figure C.2: *Illustration of the training scheme for the online AdaBoost, where only one training sample is used only for one iteration.*

A selector is able to switch between its weak classifiers so that only one of them is used for classification. Obviously, the weak classifier used should be the best of them. A selector can be seen as a meta weak classifier, which forwards the features to the best of its classifiers. Therefore, for each selector, a weight value is assigned like the weak classifiers in the original AdaBoost algorithm. The result of introducing selectors is a reduction of complexity, since only the best of the M weak classifiers per selector is evaluated.

The procedure of updating the online boosting for feature selection is as follows: First, the N selectors are initialized with each M weak classifiers. This is done randomly from the pool of weak classifiers, where each classifier makes use of one feature element in the feature vector. A training sample $(\mathbf{x}, y)$ is extracted and its sample weight, λ_n, is initialized to 1 ($\lambda_0 = 1$). The sample is then used for updating the N selectors, and is classified by each of them. Between each classification, the weight of the sample is updated according to the success of the classification. The sample weight is calculated as expressed in the following equation.

$$\lambda_n = \lambda_{n-1} \cdot \begin{cases} \frac{1}{2(1-\hat{e}_n)} & if h_n^{sel}(\mathbf{x}) = y \\ \frac{1}{2(\hat{e}_n)} & if h_n^{sel}(\mathbf{x}) \neq y \end{cases} \quad [\cdot]$$

where:

$\hat{e}_n$: Error of n-th selector. [·]
h_n^{sel}: Selector n which switches between M weak classifiers. [·]
$\mathbf{x}$: Feature vector for the trainging sample. [·]
y: The class, which $\mathbf{x}$ belongs to. [·]

The error of a selector is the ratio between the sum of the sample weights of all samples misclassified by the selector and the sum of all sample weights classified by the selector. The selector error is remembered and continuously updated with all training samples shown to the classifier. It is calculated by:

$$\hat{e}_n = \frac{\lambda_n^{wrong}}{\lambda_n^{wrong} + \lambda_n^{corr}} \quad [\cdot]$$

where:

$\hat{e}_n$: Error of n-th selector. [·]
λ_n^{wrong}: Sum of all wrongly classified sample weights by selector n. [·]
λ_n^{corr}: Sum of all correctly classified sample weights by selector n. [·]

The weight sums of correctly and wrongly classified samples are updated after updating each selector with:

$$\lambda_n^{wrong} = \lambda_n^{corr} + \lambda_{n-1} \quad [\cdot]$$
$$\lambda_n^{corr} = \lambda_n^{wrong} + \lambda_{n-1} \quad [\cdot]$$

where:

λ_{n-1}: Sample weight after updating $n-1$ selectors.
λ_n^{wrong}: Sum of all wrongly classified sample weights by selector n.
λ_n^{corr}: Sum of all correctly classified sample weights by selector n.

As in the original AdaBoost framework, each selector gets a voting weight, which expresses how much the strong classifier should depend on each selector. This is calculated from the error of each selector:

$$\alpha_n = \frac{1}{2} \log \frac{1 - \hat{e}_n}{\hat{e}_n} \quad [\cdot]$$

where:

α_n: the voting weight for selector n. [·]

The algorithm for updating an online learned AdaBoost for feature selection is described in Algorithm 3.

Algorithm 3 Algorithm for training the Online AdaBoost for feature selection.

Require: One training sample $(\mathbf{x}, y), y \in -1, +1$
Require: selector precisions $\lambda_n^{corr}, \lambda_n^{wrong}$ (initialized to 1)
$\lambda_0 \leftarrow 1$
{For all selectors}
for $n = 1, \ldots, N$ **do**
{For all weak classifiers in selector h_n^{sel} }
for $m = 1, \ldots, M$ **do**
{Update weak classifier $h_{n,m}^{weak}$ }
$h_{n,m}^{weak} = \text{update}(h_{n,m}^{weak}, (\mathbf{x}, y))$
{Update classifier error}
if $h_{n,m}^{weak}(\mathbf{x}) = y$ **then**
$\lambda_{n,m}^{corr} = \lambda_{n,m}^{corr} + \lambda_{n-1}$
else
$\lambda_{n,m}^{wrong} = \lambda_{n,m}^{wrong} + \lambda_{n-1}$
end if
$\hat{e}_{n,m} = \frac{\lambda_{n,m}^{wrong}}{\lambda_{n,m}^{wrong} + \lambda_{n,m}^{corr}}$
end for
{Select the best classifier}
$m^+ = \arg\min_m(e_{n,m})$
$e_n = e_{n,m^+}$
$h_n^{sel} = h_{n,m^+}$
{Calculate voting weight}
$\alpha_n = \frac{1}{2} \log \frac{1-\hat{e}_n}{\hat{e}_n}$
{Update sample weight}
if $h_n^s el(\mathbf{x}) = y$ **then**
$\lambda_n = \lambda_{n-1} \cdot \frac{1}{2(1-\hat{e}_n)}$
else
$\lambda_n = \lambda_{n-1} \cdot \frac{1}{2(\hat{e}_n)}$
end if
end for

When using the strong classifier, $H(\mathbf{x})$ for classification, it uses all the trained selectors and their voting weights. The strong classifier is given by:

$$H(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n \cdot h_n^{sel}(\mathbf{x}) \qquad [\cdot]$$

where:

α_n: The voting weight for selector n. [·]
$\mathbf{x}$: Extracted feature vector which needs to be classified. [·]
$h_n^{sel}(\cdot)$: Selector n. [·]
N: Total number of selectors. [·]

The strong classifier returns the raw output of the classifier. This means that it expresses the degree of which the samples resemble the target, which the classifier has been trained for. If the

classifier is to be used as a binary classifier (i.e. to express if the sample is or not the target) the sign of the raw output is used. If the output sign is positive, the sample is classified as the target and if negative, the sample is not classified as the target.

In Figure C.3 the raw output from the classifier is shown. The classifier has been learned with positive samples from detections of the person marked with a blue square over 20 frames. In the figure, the classifier is evaluated on samples extracted at every 10'th pixel with the size of the marked person and the raw output is color coded.

Figure C.3: *The raw output of the classifier.*

C.2 Features

In the previous section, the online learned AdaBoost framework was described. In this section, the features used by the classifier in this work are described. The features are the ones that Breitenstein et al. have found to yield the best results. They are color histogram, where the colors are transformed into the R'G'I color space and the Local Binary Pattern (LBP). Both features are extracted from all image patches that the tracker need to classify.

To calculate the color histogram for an image patch, all pixels are first transformed into the RGI color space. The RGI color space was proposed by Alshamasin et al. and is a computationally fast alternative to other color models where the pixel intensity is separated from the color like HSI.[28] The benefits for separating the intensity from the colors is to make the colors independent of the intensity. The transformation is given by:

$$R' = \frac{R \cdot 256}{R+G+B} \qquad [\cdot]$$

$$G' = \frac{G \cdot 256}{R+G+B} \qquad [\cdot]$$

$$I = \frac{R+G+B}{3} \qquad [\cdot]$$

Where:

R: Red component.	$[\cdot]$
G: Green component.	$[\cdot]$
B: Blue component.	$[\cdot]$
R': Transformed R color component.	$[\cdot]$
G': Transformed G color component.	$[\cdot]$
I: Intesity level.	$[\cdot]$

In contrast to the transformation from RGB space to HSI space, this transformation is fast to compute. This makes it relevant for image processing tasks, where computation time is important like for this task, where the transformation is done for many pixels in several image patches.

After the pixels of an image patch have been transformed to RGI space, the color histogram can be calculated. Breitenstein et al. have found that the best result is achieved when three bins for each channel is used. This makes color histogram consist of nine bins.

The other features used by Breitenstein et al. is the LBP, described in [47]. LBP describes the gray-level pattern in the neighborhood of a point in an image. This is repeated for every pixel in an image patch and a histogram is accumulated for the different patterns. The histogram then becomes a discriminative feature for the texture in the image patch.

Before the LBP histogram is calculated, the image patch is converted to gray-scale. For each pixel in the image patch, a sample set is extracted:

$$\mathbf{d}_{(x,y)} = [g_c \;\; g_1 \;\; g_2 \;\; g_3 \;\; g_4] \qquad [\cdot]$$

where:

$\mathbf{d}_{(x,y)}$: Sample set for pixel (x, y).	$[\cdot]$
g_c: The gray level at pixel (x, y).	$[\cdot]$
$g_1..g_4$: The gray levels of the four-connected pixels of pixel (x, y).	$[\cdot]$

The LBP descriptor is calculated by subtracting the neighbor pixels ($g_1...g_4$) with the center value (g_c) and thresholding them:

$$\mathbf{f}_{(x,y)} = [\text{sign}(g_1 - g_c) \;\; \text{sign}(g_2 - g_c) \;\; \text{sign}(g_3 - g_c) \;\; \text{sign}(g_4 - g_c)] \qquad [\cdot]$$

where:

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \text{ : Is the threshold function.} \qquad [\cdot]$$

This produces a vector with ones or zeros indicating if the neighbor values around the center pixel are higher or lower that the center value. Instead of representing the LBP as a vector, it can be summarized with as an integer:

$$LBP_{(x,y)} = \sum_{p=1}^{4} \mathbf{f}_{(x,y)}(p) 2^p \qquad [\cdot] \quad \text{(C.1)}$$

In order to summarize the LBP content of an image patch, a histogram is accumulated for all LBP integers calculated from all pixels in the image. The number of bins in the histogram is 16, corresponding to the number of possible different LBP patterns.

The Color and LBP histograms are assembled into one feature vector denoted as $\mathbf{x}$.

C.3 Weak Classifiers

The online learned AdaBoost algorithm needs weak classifiers that are able to be updated online. In this section, the weak classifiers used in this work are described. They are the ones Grabner et al. proposed to use on histogram features. [42]

Each weak classifier makes use of one feature element, so that the number of weak classifiers is the same as the number of features in the feature vector denoted by $\mathbf{x}$. Each weak classifier estimates two Gaussian distributions from the samples it is updated with. One for all positive samples and one for the negative. When classifying a sample, the two Gaussian distributions are evaluated and the one that yields the highest value is the class, which the sample is classified as. This is illustrated in Figure C.4.

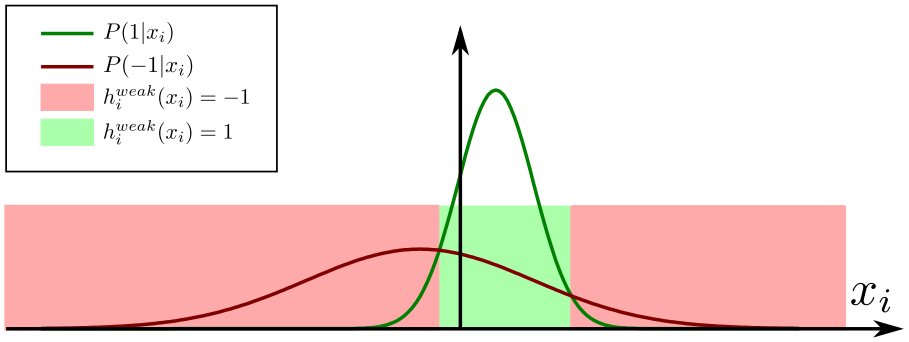


Figure C.4: *The two estimated Gaussian distributions from positive and negative samples along with the outcome of the weak classifier given a x_i.*

Since the weak classifiers are supposed to work in an online framework, the distributions for the positive and negative class should be updated online. For this, a scalar Kalman filter is used. The scalar Kalman state space model is given by:

$$z_t = A \cdot z_{t-1} + B \cdot u_{t-1} + w_{t-1} \quad [\cdot] \quad \text{(C.2)}$$

$$y_t = C \cdot z_t + v_t \quad [\cdot] \quad \text{(C.3)}$$

where:

z_t: State varible at time t. [·]

A: Constant for transfering the state to next timestep. [·]

B: Constant for the relation between control input and state. [·]

C: Constant for the relation between state and measurement. [·]

u_t: Control input at time t. [·]

y_t: Measurement at time t. [·]

v_t: Measurement noise: zero mean Gaussian noise with variance R. [·]

w_t: Process noise: zero mean Gaussian noise with variance Q. [·]

The prediction of the state in the next time step given the state in current time step can be calculated by:

$$z_t^* = A \cdot z_{t-1} + B \cdot u_{t-1} \quad [\cdot] \quad \text{(C.4)}$$

Furthermore, a predicted error for this a priori estimate can be calculated by:

$$P_t^* = A^2 P_{t-1} + Q \quad [\cdot]$$

When a new measurement (y_t) arrives at the system, the predicted state can be updated with the follwing equations:

$$z_t = z_t^* + K_t(y_t - C \cdot z_t^*) \quad [\cdot]$$

Where K_t is the Kalman gain or sometimes called the blending factor. It is calculated by:

$$K_t = \frac{C \cdot P_t^*}{C^2 \cdot P_t^* + R} \quad [\cdot]$$

The estimated prediction error can also be updated with the new measurement with this equation:

$$P_t = (1 - C \cdot K_t) \cdot P_t^* \quad [\cdot]$$

In order to modify the Kalman filter to estimate mean and variance for a sequence of samples, the state space model is modified to be $\mu_t = \mu_{t-1} + v_t$ for mean and $\sigma_t^2 = \sigma_{t-1}^2 + v_t$ for variance. To obtain this from the model in Equation C.2 and C.3, the parameters are set to $A = 1$, $B = 0$, $C = 1$, $Q = 0$ and $R = 0.01$. [42] This results in the prediction step being reduced to:

$$\mu_t^* = \mu_{t-1} \quad [\cdot]$$
$$\sigma_t^{*2} = \sigma_{t-1}^2 \quad [\cdot]$$
$$P_t^* = P_{t-1} \quad [\cdot]$$

Since the models for μ and σ^2 are defined to have the same source of noise, they have a common kalman gain (K_t) and error (P_t). The update equations are then derived to be:

$$\begin{aligned} K_t &= \frac{P_t^*}{P_t^* + R} \\ &= \frac{P_{t-1}}{P_{t-1} + R} \quad [\cdot] \\ \mu_t &= \mu_t^* + K_t(x_t - \mu_t^*) \\ &= \mu_{t-1} + K_t(x_t - \mu_{t-1}) \quad [\cdot] \\ \sigma_t^2 &= \sigma_t^{*2} + K_t((x_t - \mu_t)^2 - \sigma_t^{*2}) \\ &= \sigma_{t-1}^2 + K_t(x_t - \sigma_{t-1}^2) \quad [\cdot] \\ P_t &= (1 - K_t) \cdot P_t^* \\ &= (1 - K_t) \cdot P_{t-1} \quad [\cdot] \end{aligned}$$

where:

x_t: The feature element at time t to estimate mean and variance for. $[\cdot]$

Before the Kalman filter can start, the parameters $P_0, \mu_0 and \sigma_0^2$ needs to be set. Grabner et al. propose to initialize them to $P_0 = 1000, \mu_0 = 0, \sigma_0^2 = 0$.

Appendix D

Particle Filter

When tracking objects in practice, the measured object positions will contain noise. To compensate for this, estimation techniques that model the state of an object (e.g. position and velocity) using both a measurement (typically the image coordinates of the object) and the uncertainties of the model, have been widely used. The Kalman filter has been a popular estimation technique for object tracking. It works under a set of assumptions. These assumptions include that the system must be linear; the initial state must be Gaussian; the noise that the measurements are subject to must be white and Gaussian (i.e. the state is Gaussian distributed).To overcome the assumption of the system being linear, the Extended Kalman filter was proposed. The Extended Kalman filter also assumes that the state is Gaussian distributed. Given that the aforementioned assumptions hold, the Kalman filter yields the optimal estimate. Tracking an object using the Kalman filter might work satisfactorily as long as the object is not occluded. However, under occlusion, the pdf of the object state variables becomes multi-modal, i.e. non-Gaussian. In this case, the Kalman filter will give poor estimates. A particle filter overcomes the limitation of the state variables having to be Gaussian distributed, which has made it a popular technique for object tracking. In this appendix, one variant of particle filters, which this work is based on, i.e. the Condensation algorithm, is described. [26] [48]

To estimate an object state $\mathbf{x}$ from a measurement $\mathbf{z}$, the posterior density $p(\mathbf{x}|\mathbf{z})$ must be evaluated, as it represents all the knowledge about the state $\mathbf{x}$ that can be extracted from the data. The posterior density can be calculated using Bayes' formula as expressed in Equation D.1.

$$p(\mathbf{x}|\mathbf{z}) \propto p(\mathbf{z}|\mathbf{x}) \cdot p(\mathbf{x}) \qquad [\cdot] \quad \text{(D.1)}$$

where:

$\mathbf{x}$: A state vector. $[\cdot]$

$\mathbf{z}$: A measurement vector. $[\cdot]$

As it can be seen in Equation D.1, the conditional density $p(\mathbf{z}|\mathbf{x})$, which expresses the likelihood of possible states leading to the given measurement, must be calculated. However, in cases, where the conditional density $p(\mathbf{z}|\mathbf{x})$ is multi-modal, it can become too complex to calculate explicitly due to a large state-space. Therefore, in the general case, it must be estimated through iterative sampling. One technique for this is the Condensation algorithm.

D.1 Condensation

The output of each iteration in the Condensation algorithm is a weighted sample-set $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)}), n = 1, 2, \ldots, N\}$, which works as an approximation of the conditional density $p(\mathbf{x}|\mathbf{z})$ at time-step t.

The principle of the Condensation algorithm is illustrated in Figure D.1, where one time-step is shown.

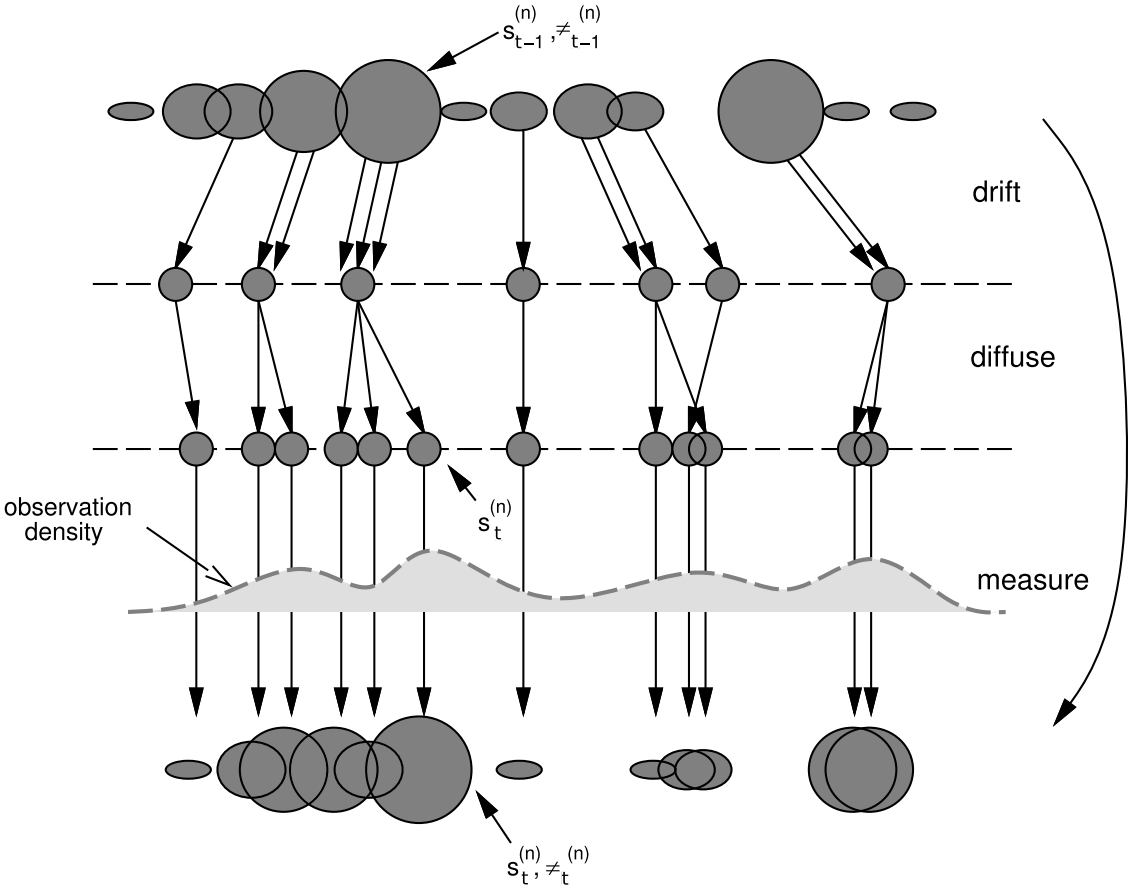


Figure D.1: *One time-step in the Condensation algorithm.* $s_{k-1}^{(n)}$ *and* $\pi_{k-1}^{(n)}$ *respectively denote sample* n *and its corresponding weight, both at time* $(k-1)$*. Image courtesy [48]*

The first step is to sample from the set at time (t-1). Each sample is chosen with probability $\pi_{t-1}^{(n)}$. The chosen samples are then diffused and drifted using a motion model. At the final step, weights of the samples are estimated using an observation model. A synopsis of the algorithm is given in Figure D.2.

D.1.1 Initial Conditions

To initialize the algorithm, the initial conditions must be determined. Provided that the prior density $p(\mathbf{x}_0)$ is Gaussian, the algorithm can be initialized using direct sampling.

D.1.2 Data Association

When tracking multiple objects, a technique for associating the most likely measurement with a state of a particular object is necessary. As described in the "Object Tracking" section (section 2.5.1) in the Analysis chapter, several techniques for this exist, like nearest neighbor approaches, greedy search, JPDAF, MHT and PMHT.

Iteration

From the sample set $\{(\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}), n = 1, 2, \ldots, N\}$ at time $(t-1)$, the sample set for time t; $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)}), n = 1, 2, \ldots, N\}$ is constructed.

Sample n of N new samples is selected using the following scheme:

1. **Selection**: Select the n^{th} sample, $\hat{s}_t^{(n)}$:
   - **(a)** select a random number $r \in [0, 1]$.
   - **(b)** find the smallest j so that $c_{t-1}^{(j)} > r$.
   - **(c)** set $\hat{\mathbf{s}}_t^{(n)} = \mathbf{s}_{t-1}^{(j)}$.
2. **Prediction**: Predict by sampling from $p(\mathbf{x}_t | \mathbf{x}_{t-1} = \hat{\mathbf{s}}_t^{(n)})$. The new sample value may be generated using a motion model.
3. **Correction**: Calculate and store the weight of the sample together with a cumulative probability.
   - **(a)** Calculate the weight, $\pi_t^{(n)}$ of $\mathbf{s}_t^{(n)}$. This may be done by estimating $p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{s}_t^{(n)})$ using an observation model.
   - **(b)** Normalize the weights so that $\sum_n \pi_t^{(n)} = 1$.
   - **(c)** Store the weight together with a cumulative probability, $c_t^{(n)} = c_t^{(n-1)} + \pi_t^{(n)}$; $c_t^{(0)} = 0$.

After constructing the N samples, moments of the tracked object position can be estimated as

$$\mathcal{E}[f(\mathbf{x}_t)] = \sum_{n=1}^{N} \pi_t^{(n)} \cdot f\left(\mathbf{s}_t^{(n)}\right),$$

which can be used e.g. to estimate a mean position for visualization.

Figure D.2: *The Condensation Algorithm. [48]*

Appendix E

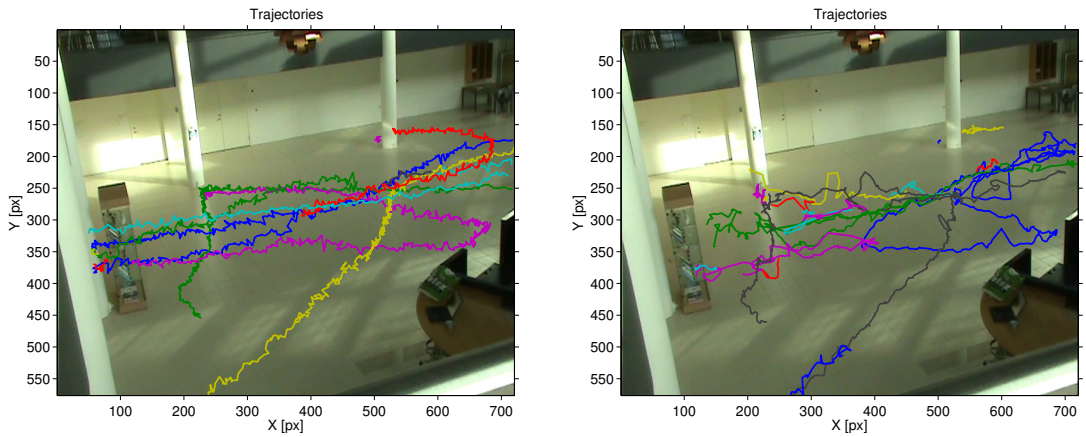
Visualization Experiment Plots

In this appendix, the plots from the "Visualization" module experiment are shown. Each figure shows the estimated plot and the GT plot of the statistic in question and is followed by a qualitative comparison between these plots. First, the plots from the AAU1 dataset are shown and next, the plots from Friis1 are shown. Before the statistic plots from AAU and Friis, the corresponding GT and the estimated trajectories are shown. Although these plots are not part of the evaluation, they are shown to give an impression of the data used to generate the following statistics. Furthermore, the estimated trajectories can be found on the CD in the back of this report.

E.1 AAU

In Figure E.1a and E.1b, the GT and the estimated trajectories are plotted after the trajectory bounding boxes have been converted to feet coordinates. The trajectories are plotted on a empty frame from the AAU1 video sequence.

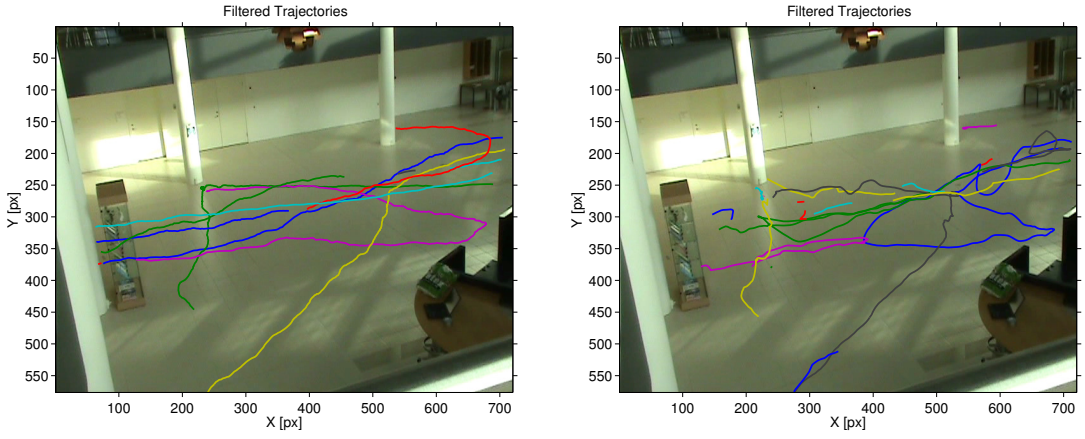
It is seen in Figure E.1a how the GT trajectories have low frequency noise on trajectories. This noise is filtered out by the filter, and the resulting trajectories can be seen in Figure E.2a.



(a) **(b)**

Figure E.1: *(a): GT trajectories plotted on an empty frame from the AAU1 video sequence. (b): Estimated trajectories for the same video sequence are plotted.*

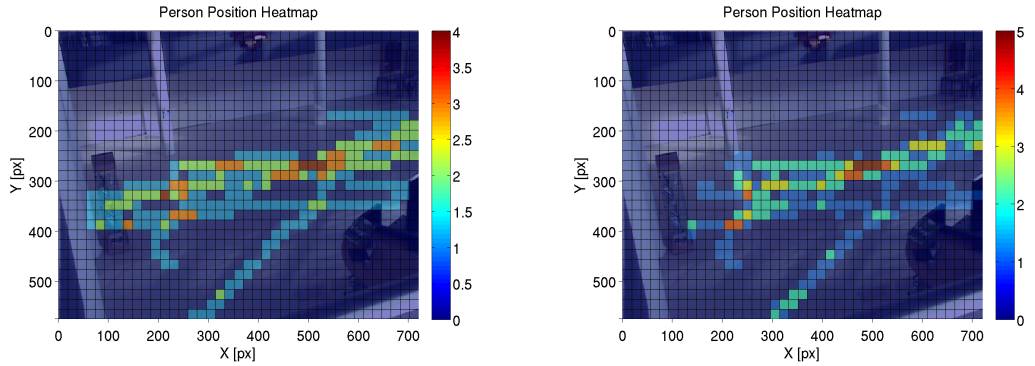
In Figure E.2a and E.2b, the estimated trajectories and GT trajectories are plotted after filtering.



(a) **(b)**

Figure E.2: *(a): The GT trajectories plotted in Figure E.1a after filtering. (b): Filtered estimated trajectories from Figure E.1b.*

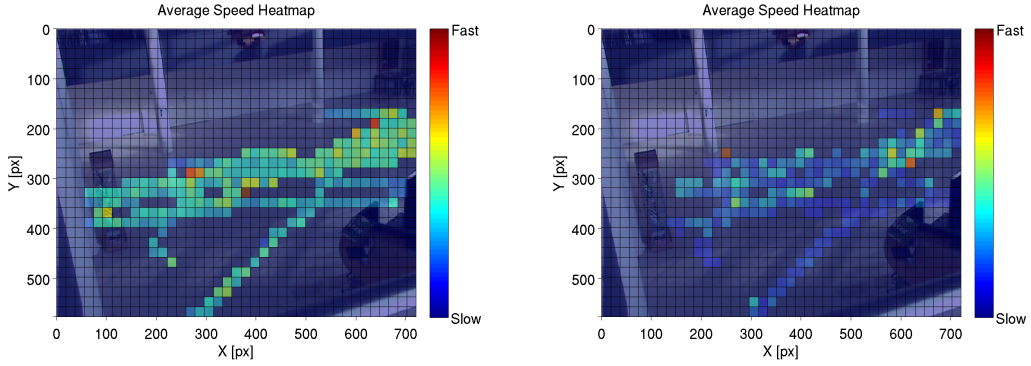
In Figure E.3a and E.3b, the Person Position heat maps are shown. The heat maps have been calculated from the filtered feet trajectories from the GT and the estimated data. It is seen that near (520,280) in the GT heat map, a high person count occurs, which matches the heat map generated on estimated trajectories. Furthermore, this agrees with what can be seen in the plot with trajectories in Figure E.2, where many different persons are passing through those bins. It can also be seen that the two Person Position heat maps resemble each other, although they are not identical.



(a) **(b)**

Figure E.3: *Person Position heat maps. (a) Heat map generated from GT trajectories. (b): Heat map generated from estimated trajectories.*

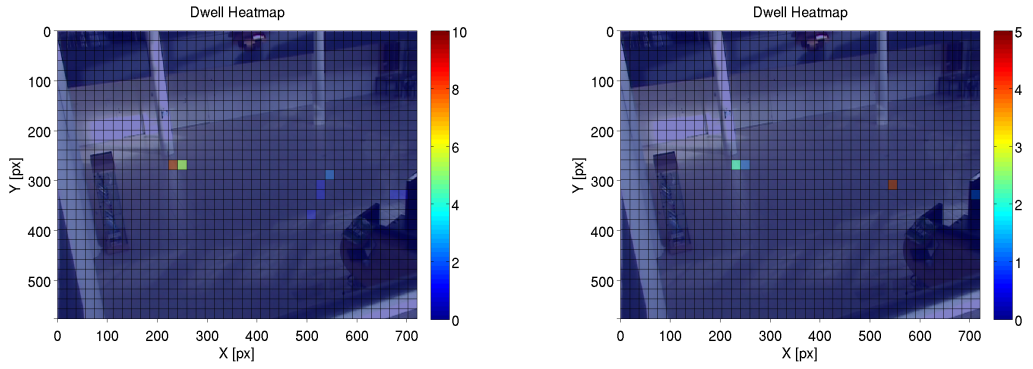
In Figure E.4a and E.4b, heat maps that illustrate the Speed of people are shown. It is seen how the overall color of the estimated heat map is closer to blue than the overall color of the GT heat map. This is not caused by the Speed being estimated to be lower, but rather by one or two noisy trajectories. Therefore, they yield a high speed resulting in the average speed of the bin that it belongs to be high and thus expanding the scale of the colors. Apart from the overall color level being different, the two heat maps have similar shapes.



(a) **(b)**

Figure E.4: *Speed heat map. (a): Heat map generated from GT trajectories. (b): Heat map generated from estimated trajectories.*

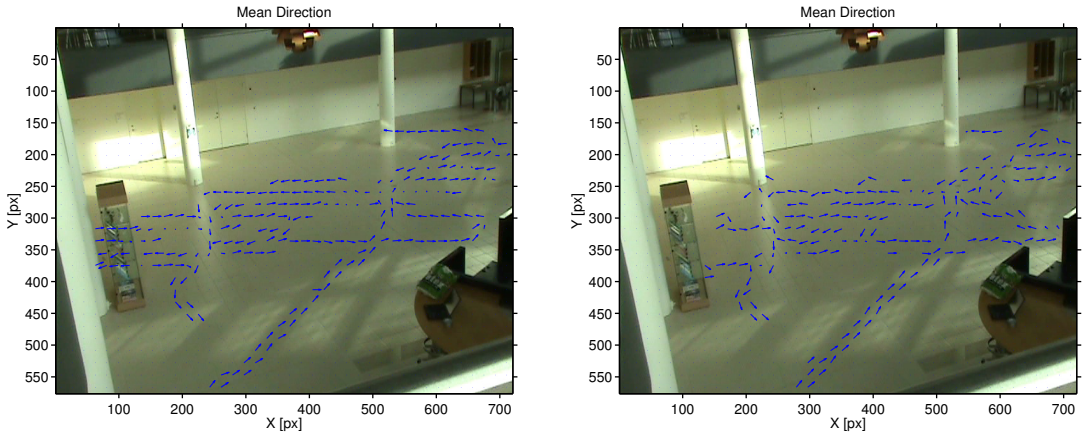
In the Figures E.5a and E.5b, heat maps of where people tend to dwell are shown. The GT plot shows dwell activity in front of the left pole. This matches with what happens in the video; a person stops up and looks at the pole. This dwell is also recorded on the dwell heat map generated from the estimated trajectories. The dwells in the right of the GT heat map is caused by a person stopping up and turning around. Part of this is also shown in the estimated heat map.



(a) **(b)**

Figure E.5: *Dwell heat map. (a): Heat map generated from GT trajectories. (b): Heat map generated from estimated trajectories.*

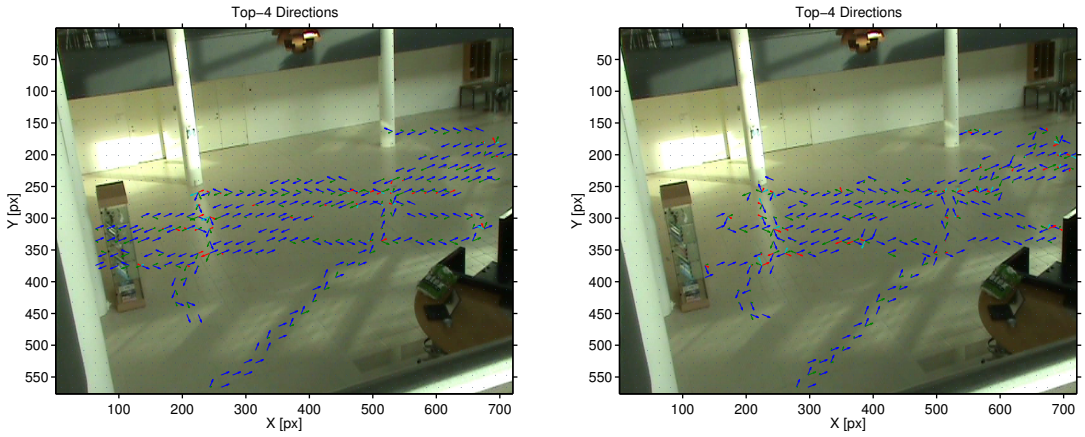
In Figure E.6a, the Mean Directions of the movement of peoplefor each bin are illustrated and have been generated from GT trajectories. It agrees with what can be seen in the video sequence. For instance, in the bins near row 340-350 and columns 120-220, two persons are passing through in opposite directions. This results in the mean direction being neutralized, which is illustrated by short arrows. In bins, where only one person is passing through or multiple persons passing in the same direction, the arrows are longer, which indicates a consistent flow direction. In Figure E.6a, the same direction plot is shown for estimated trajectories. It is seen that the two heat maps resemble each other.



(a) **(b)**

Figure E.6: *Mean Direction plot. (a): Plot generated from GT trajectories. (b): Plot generated from estimated trajectories.*

In Figure E.7b, a plot with the Top-Four Directions is shown, generated from GT trajectories. The same examples for validating the mean Directions plot can be used to validate this Top-Four Directions plot. Bins, where people are passing opposite have arrows pointing in each directions, while bins where one or more persons are passing in the same direction have most frequent directions pointing almost the same directions.



(a) **(b)**

Figure E.7: *Top-Four Directions plot. (a): Plot generated from GT trajectories. (b): Plot generated from estimated trajectories.*

In Figure E.8, the number of people which the "Multi-Human Tracker" has estimated to be in the FOV of the camera is shown. For an easier comparison, the GT has been plotted on the same figure.

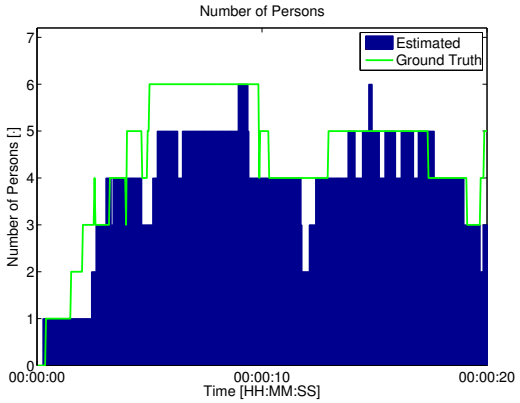


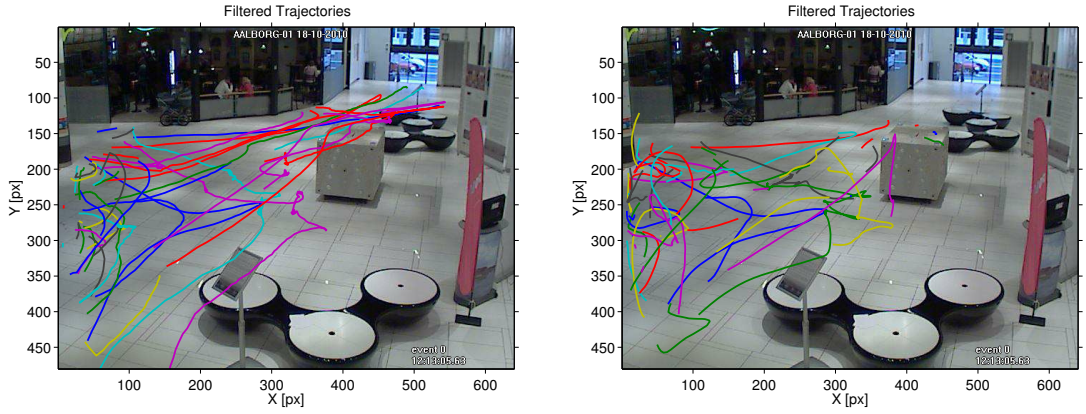
Figure E.8: *Number of Persons plot, with both the plot generated from estimated trajectories and the plot generated from GT trajectories.*

It is seen that the estimated number of persons is generally one person away from the GT.

E.2 Friis

In this section, the plots from the Friis1 dataset are shown.

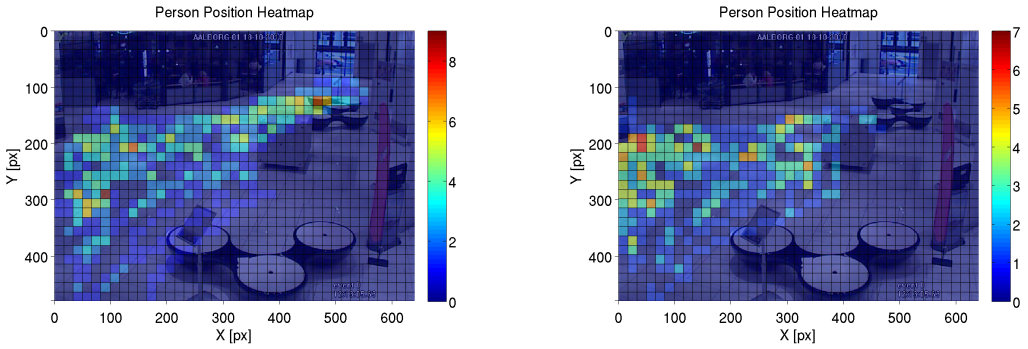
In Figure E.9a and E.9b the estimated trajectories and the GT trajectories are shown after filtering.



(a) **(b)**

Figure E.9: *GT and estimated trajectories. Figure (a) shows the filtered GT trajectories. Figure (b) shows the filtered estimated trajectories.*

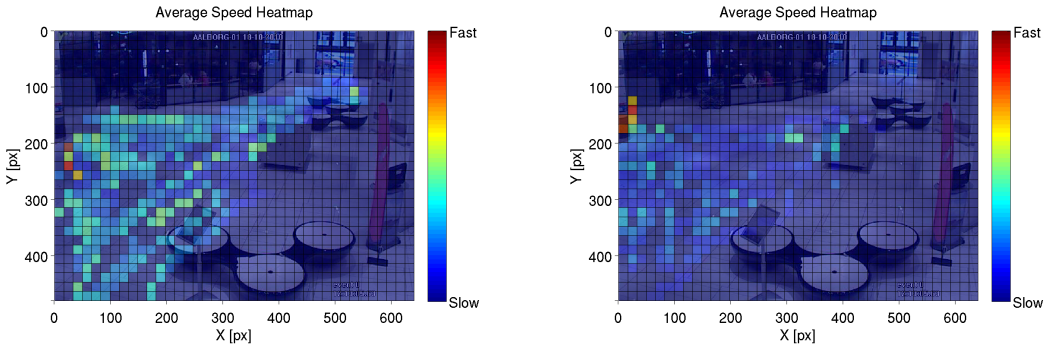
In Figure E.10a and E.10b, the Person Position heat maps are shown. It is seen that the overall tendencies for where persons walk is the same between the estimated and the GT. However, the bins above the 150 y-coordinate have values close to zero in the estimated plots. This is caused by the people in those areas being too small to get detected by the system.



(a) **(b)**

Figure E.10: *(a): GT person position heat map. (b): Estimated Person Position heat map.*

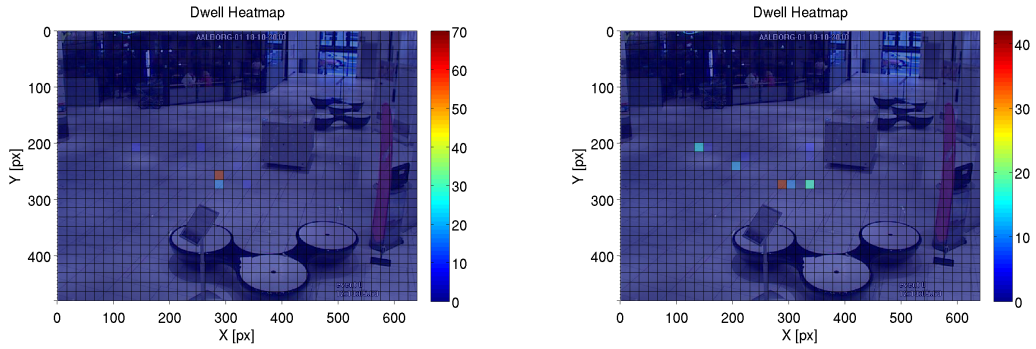
In Figure E.11a and E.11b, the heat maps of the speed of people are shown. In the estimated plot, high values between y-coordinate 100 and 200 are present to the left of the frame. This is caused by the trackers following persons that leave the scene being terminated too slowly. As a consequence of this, a lost tracker shuffles around until it is terminated. This yield high values in that area, since many people left the scene and thus extends the color scale to stretch over greater range making visual comparison difficult.



(a) **(b)**

Figure E.11: *(a): GT Average Speed heat map. (b): Estimated Average Speed heat map.*

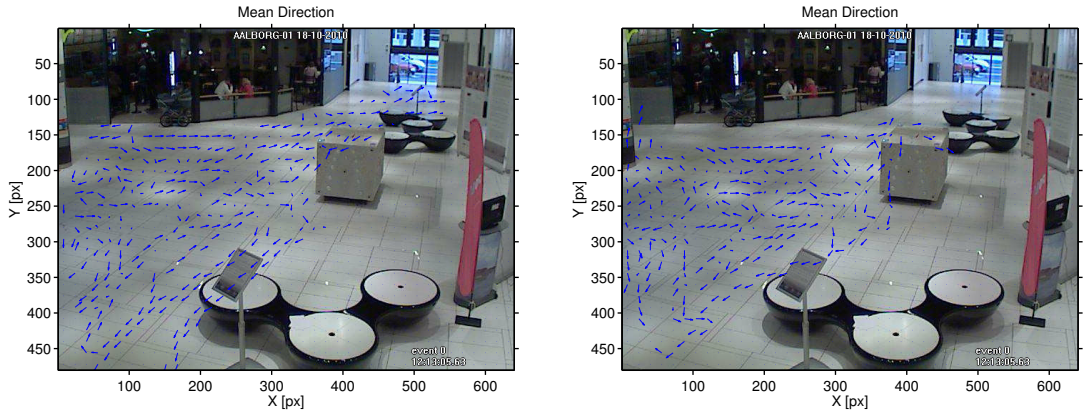
In the Figures E.12a and E.12b heat maps of where people tend to dwell are shown. It is seen how a high number of dwells is observed for the bin at approximately (280,350). This is roughly the same in the estimated plot. Furthermore, small numbers of dwells are scattered around elsewhere in the estimated plot, which are correct for some.



(a) **(b)**

Figure E.12: *(a): GT dwell heat map. (b): Estimated dwell heat map.*

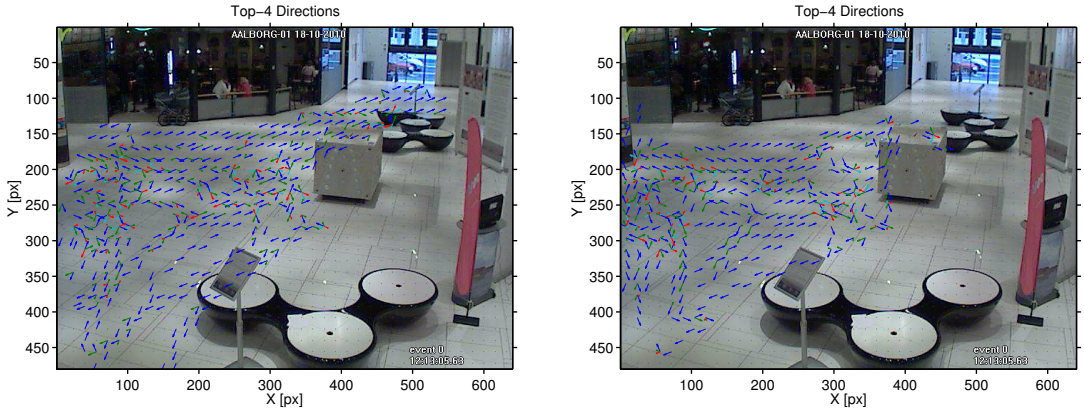
In Figure E.13, the Mean Directions are shown as vector plots. The estimated and GT plots generally resemble each other, with small errors occuring.



(a) **(b)**

Figure E.13: *(a): GT Mean Direction plot. (b): Estimated Mean Direction plot.*

In Figure E.14, the Top-Four Directions are shown as vector plots. Like the Mean Directions, the estimated and GT plots resembles each other, with small errors present.



(a) **(b)**

Figure E.14: *(a): GT Top-Four Direction plot. (b): Estimated Top-Four Direction plot.*

In Figure E.15 the number of people inside the FOV of the cameras is shown. For easier comparison the GT has been plotted on the same figure. The difference between the GT curve and the estimated number of persons is varying throughout the sequence. At two points, the different can be considered so be considered significantly high. The first is at approximately 12:14:30. Here, the amount of people is underestimated. This is caused by many people entering the scene at the same time and at the same time resulting in extensive occlusions. The same happens when the system underestimates at approximately 12:15:15.

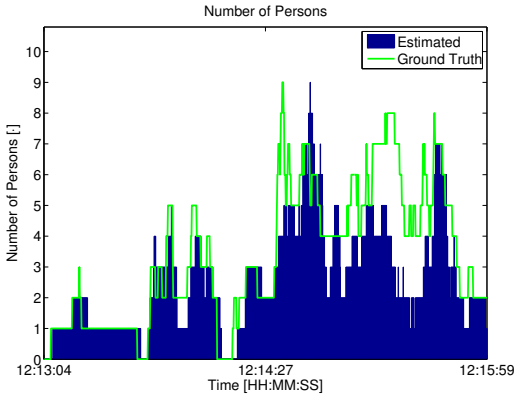


Figure E.15: *Number of Persons plot generated from estimated trajectories and GT trajectories.*

Bibliography

[1] A. Bjerregaard, "Danskerne vil have tryghed gennem overvågning," *http://www.altinget.dk/artikel.aspx?id=91148*, 2009.

[2] J. Gudmundsson, P. Laube, and T. Wolle, "Movement Patterns in Spatio-temporal Data," in *Encyclopedia of GIS*, 2008, pp. 726–732.

[3] T. S. Nielsen, H. Harder, and P. N. Jensen, "Detecting user patterns in public green space: A GPS based survey of people's behavior and movement patterns in a park in Denmark." World in Denmark, 2010.

[4] Axis Communications, "Intelligent video applications," *http://www.axis.com/products/video/about_networkvideo/iv/products.htm*.

[5] D. Hedegaard, "Videoovervågning - En teoretisk og empirisk diskussion af virkningsforståelser," Master's thesis, Aarhus Universitet, 2008.

[6] Nordjyllands Politi, "Evaluering af TV-Overvågningen af Jomfru Ane Gade I Aalborg," *http://www.politi.dk/Nordjylland/da/lokalnyt/Nyheder/Evaluering+af+TV-overv%C3%A5gning+ved+Jomfru+Ane+Gade+i+Aalborg030409.htm*, 2009.

[7] S. Wojcicki, "Making ads more interesting," *http://googleblog.blogspot.com/2009/03/making-ads-more-interesting.html*, 2009.

[8] R. R. Burke, "The Third Wave of Marketing Intelligence," in *Retailing in the 21st Century*, M. Krafft and M. K. Mantrala, Eds. Springer Berlin Heidelberg, 2010, pp. 159–171.

[9] BBC News, "How we are being watched," *http://news.bbc.co.uk/2/hi/uk_news/6110866.stm*, 2006.

[10] DR, "50 overvågningskameraer op i timen," *http://www.dr.dk/Nyheder/Indland/2008/04/22/095832.htm*, 2008.

[11] Forbrugerrådet, "100.000 overvågningskameraer i danmark," *http://www.forbrugerraadet.dk/nyheder-alle/overvaagning/?ref=2820*, 2004.

[12] G. Orwell, *Nineteen Eighty-Four*. Secker and Warburg, 1949, ISBN-13: 978-0-436-41055-0.

[13] T. Breinstrup, "International rapport: Udbredt overvågning i Danmark," *http://www.business.dk/tech-mobil/international-rapport-udbredt-overvaagning-i-danmark*, 2008.

[14] Politiken, "Danskerne elsker overvågning," *http://politiken.dk/indland/ECE302494/danskerne-elsker-overvaagning/*, 2007.

[15] K. Schultz, "The Uses of the Tally Counter," *http://www.ehow.com/list_7660596_uses-tally-counter.html*, 2010.

[16] SenSource, "Learn About Our Sensing and Data Collection Technology," *http://www.sensourceinc.com/technology.htm*.

[17] Axiomatic, "How Many People Exactly?" *http://www.airport-int.com/article/flow-rate-information.html*, 2010.

[18] Traf-Sys Inc., "Thermal Sensor - Customer Counting Systems," *http://www.trafsys.com/people-counters/thermal-sensor.aspx*.

[19] Thermitrack, "Thermitrack Product Information," *http://www.thermitrack.com/prodinfo.html*.

[20] E. Murphy-Chutorian and M. M. Trivedi, "Head Pose Estimation in Computer Vision: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 607–626, April 2009.

[21] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. V. Gool, "Online Multi-Person Tracking-by-Detection from a Single, Uncalibrated Camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010, in press.

[22] C. H. Kuo, C. Huang, and R. Nevatia, "Multi-target tracking by Online Learned Discriminative Appearance Models," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010, pp. 685 –692.

[23] B. Babenko, M. H. Yang, and S. Belongie, "Visual tracking with online Multiple Instance Learning," *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 983 –990, 2009.

[24] J. Kerridge, S. Keller, T. Chamberlain, and N. Sumpter, "Collecting Pedestrian Trajectory Data In Realtime," in *Pedestrian and Evacuation Dynamics 2005*, N. Waldau, P. Gattermann, H. Knoflacher, and M. Schreckenberg, Eds. Springer Berlin Heidelberg, 2007, pp. 27–39.

[25] J. Candamo, M. Shreve, D. B. Goldgof, D. B. Sapper, and R. Kasturi, "Understanding transit scenes: A survey on human behavior-recognition algorithms," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 1, pp. 206–224, 2010.

[26] A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: A Survey," *ACM Comput. Surv.*, vol. 38, no. 4, p. 45, 2006.

[27] A. Agarwal and B. Triggs, "3D Human Pose from Silhouettes by Relevance Vector Regression," in *In CVPR*, 2004, pp. 882–888.

[28] M. Alshamasin, R. Al-kasasbeh, A. Khraiwish, Y. Al-shiboul, and D. E. Skopin, "Acceleration of Image Processing Using New Color Model," *Am. J. Applied Sci.*, vol. 6, no. 5, p. 6, 2009.

[29] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, p. 511, 2001.

[30] T. Ojala, M. Pietikainen, and T. Mäenpää, "Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, 2002.

[31] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886 –893 vol. 1, june 2005.

[32] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91 – 110, 2004.

[33] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *In ECCV*, 2006, pp. 404–417.

[34] G. Bradski and A. Kaehler, *Learning OpenCV*. O'Reilly Media, September 2008, ISBN-13: 978-0-596-51613-0.

[35] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 5, pp. 564 – 577, May 2003.

[36] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. John Wiley & Sons, 2000, ISBN-13: 978-0-471-05669-0.

[37] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. V. Gool, "Project page for "Online Multi-Person Tracking-by-Detection from a Single, Uncalibrated Camera"," *http://www.vision.ee.ethz.ch/~bremicha/tracking/*, 2010.

[38] B. E. Flores, "A pragmatic view of accuracy measurement in forecasting," *Omega*, vol. 14, no. 2, pp. 93–98, 1986.

[39] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The CLEAR MOT metrics," *Journal on Image and Video Processing*, vol. 2008, pp. 1–10, 2008.

[40] V. A. Prisacariu and I. Reid, "FastHOG - a real-time GPU implementation of HOG. Technical Report No. 2310/09," Tech. Rep., July 2009, Department of Engineering Science, Oxford University.

[41] N. Cristianini and J. Shaw-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. University of Cambridge, 2000, ISBN: 0-521-78019-5.

[42] H. Grabner and H. Bischof, "Online Boosting and Vision," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, 2006, pp. 260 – 267.

[43] L. N. Trefethen and D. Bau, III, *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997, ISBN-13: 978-0-898713-61-9.

[44] S. M. Ross, *Probability and Statistics for Engineers and Scientists*. Elsevier Academic Press, 2004, ISBN-13: 978-0-12-598057-9.

[45] C. Wojek, G. Dorkó, A. Schulz, and B. Schiele, "Sliding-Windows for Rapid Object Class Localization: A Parallel Technique," in *Proceedings of the 30th DAGM symposium on Pattern Recognition*. Springer-Verlag, 2008, pp. 71–81.

[46] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting," 1995.

[47] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[48] M. Isard and A. Blake, "CONDENSATION - Conditional Density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, pp. 5–28, 1998.

List of Corrections